JP2003348598

Publication Title:

METHOD AND APPARATUS FOR MEMORY EFFICIENT COMPRESSED DOMAIN VIDEO PROCESSING AND FOR FAST INVERSE MOTION COMPENSATION USING FACTORIZATION AND INTEGER APPROXIMATION

Abstract:

Abstract of JP2003348598

<P>PROBLEM TO BE SOLVED: To provide a method and apparatus for memory efficient compressed domain video processing and for fast inverse motion compensation using factorization and integer approximation. <P>SOLUTION: A method for reducing memory requirements needed to decode a bit stream comprises: receiving a video bit stream; decoding the frame of the bit stream into a discrete cosine transform (DCT) domain representation; identifying non-zero coefficients of the DCT domain representation; assembling a hybrid data structure; and inserting the nonzero coefficients of the DCT domain representation into the hybrid data structure. A method for performing inverse motion compensation is provided. The method initiates with receiving a video bit stream, and then, a transform matrix type is identified. The transform matrix type is either a half pixel matrix or a full pixel matrix. If the transform matrix type is a half pixel matrix, then the method includes applying a factorization technique to decode the bit stream corresponding to the half pixel matrix. If the transform matrix type is a full pixel matrix, then the method includes applying an integer approximation technique to decode the bit stream corresponding to the full pixel matrix. <P>COPYRIGHT: (C)2004,JPO

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2003－348598
（P2003－348598A）
(43)公開日　平成15年12月5日(2003.12.5)

(51)Int.Cl.⁷		識別記号	ＦＩ		テーマコード(参考)
Ｈ０４Ｎ	7/32		Ｈ０３Ｍ	7/36	５Ｃ０５９
Ｈ０３Ｍ	7/36			7/40	５Ｊ０６４
	7/40		Ｈ０４Ｎ	7/137	Ｚ

審査請求　未請求　請求項の数65　ＯＬ　外国語出願　（全 97 頁）

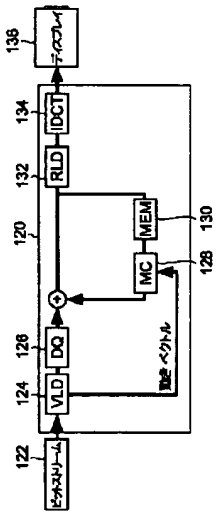
(21)出願番号　特願2003－107352(P2003－107352)
(22)出願日　平成15年４月11日(2003.4.11)
(31)優先権主張番号　１０／３１９７７５
(32)優先日　平成14年12月13日(2002.12.13)
(33)優先権主張国　米国（ＵＳ）
(31)優先権主張番号　１０／３１９７４７
(32)優先日　平成14年12月13日(2002.12.13)
(33)優先権主張国　米国（ＵＳ）
(31)優先権主張番号　６０／３７２２０７
(32)優先日　平成14年４月12日(2002.4.12)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　000002369
セイコーエプソン株式会社
東京都新宿区西新宿２丁目４番１号
(72)発明者　ウィリアム　チェン
アメリカ合衆国　カリフォルニア州　フォスタシティ　ロック　ハーバー　レーン　138
(74)代理人　100095728
弁理士　上柳　雅誉　（外２名）

最終頁に続く

(54)【発明の名称】　メモリ効率のいい圧縮領域ビデオ処理のための且つ因数分解及び整数近似法を用いる高速逆動き補償のための方法並びに装置

(57)【要約】
【課題】　メモリ効率のいい圧縮領域映像処理のための且つ因数分解及び整数近似法を用いる高速逆動き補償のための方法並びに装置
【解決手段】　ビットストリームを復号化するのに要するメモリ所要量を減らすための方法。この方法はビデオビットストリームを受け取り、ビットストリームのフレームが離散コサイン変換（DCT）領域表現に復号化される。次に、そのDCT領域表現の非零係数が識別され、ハイブリッドデータ構造がアセンブルされ、DCT領域表現の非零係数がハイブリッドデータ構造に挿入される。また、逆動き補償を実行するための方法では、ビデオビットストリームを受け取け、変換行列タイプが識別される。変換行列タイプは、半画素行列か完全画素行列かのどちらかで、変換行列タイプが半画素行列ならば、その半画素行列に対応するビットストリームを復号化する因数分解技法を適用し、変換行列タイプが完全画素行列ならば、完全画素行列に対応するビットストリームを復号化する整数近似技法を適用する。



【特許請求の範囲】
【請求項１】　ビットストリームを復号化するのに要するメモリ所要量を低減するための方法であって、
ビデオビットストリームを受け取り、
前記ビットストリームのフレームを変換領域表現に復号化し、
前記変換領域表現の非零係数を識別し、
固定サイズアレイ及び可変サイズオーバーフローベクトルを含むハイブリッドデータ構造をアセンブルし、
前記変換領域表現の非零係数を前記ハイブリッドデータ構造の中に挿入することからなることを特徴とする方法。
【請求項２】　前記ビデオビットストリームは、低レートビデオビットストリームであることを特徴とする請求項１に記載の方法。
【請求項３】　前記ビットストリームのフレームをを変換領域表現に復号化するオペレーションは、
可変長デコーダ及び反量子化ブロックを通る前記ビットストリームを処理することを含むことを特徴とする請求項１に記載の方法。
【請求項４】　前記固定サイズアレイは固定サイズブロックを含むことを特徴とする請求項１に記載の方法。
【請求項５】　前記固定サイズブロックは、前記変換領域表現の８つの非零係数を保持するように構成されていることを特徴とする請求項４に記載の方法。
【請求項６】　前記変換領域表現の非零係数をハイブリッドデータ構造に挿入するオペレーションは、
フレームのブロック毎に、固定サイズアレイの中の係数を可変サイズオーバーフローベクトルの中の対応する係数に写像することを含むことを特徴とする請求項１に記載の方法。
【請求項７】　ビデオデータを復号化するための方法であって、
圧縮ビットストリーム内のビデオデータのフレームを受け取り、
前記フレームのブロックを圧縮領域で変換領域表現に復号化し、
ハイブリッドデータ構造を定義し、
前記変換領域表現と関連付けられるデータを前記ハイブリッドデータ構造に保持し、
前記変換領域表現と関連付けられるデータに逆動き補償を前記圧縮領域で実行し、
前記データに逆動き補償を実行した後、表示するためにデータを解凍することからなることを特徴とする方法。
【請求項８】　前記ハイブリッドデータ構造は、固定サイズブロックからなる固定サイズアレイ及び可変サイズオーバーフローベクトルを含んでいることを特徴とする請求項７に記載の方法。
【請求項９】　前記変換領域表現と関連付けられるデータをハイブリッドデータ構造に保持するオペレーションは、
前記変換領域表現の非零係数を識別し、
前記固定サイズブロックの容量に達するまで、前記非零係数を前記ハイブリッドデータ構造の固定サイズアレイの固定サイズブロックの中に保持し、
前記固定サイズブロックの容量に達した後は、固定サイズブロックの容量を超える非零係数をオーバーフローベクトルに保持することを含むことを特徴とする請求項７に記載の方法。
【請求項１０】　前記圧縮ビットストリームは、低レートビットストリームであることを特徴とする請求項７に記載の方法。
【請求項１１】　前記変換領域表現と関連付けられるたデータに逆動き補償を圧縮領域で実行するオペレーションは、
前記変換領域表現と関連付けられるデータに、ハイブリッド因数分解及び整数近似技法を適用することを含むことを特徴とする請求項７に記載の方法。
【請求項１２】　ハイブリッドデータ構造に保持するために低レートビットストリームを並べ替えるためのプログラム命令を有するコンピュータ可読メディアであって、
データフレームの符号化されたブロックと関連付けられる非零変換係数を識別するためのプログラム命令と、
前記非零変換係数を固定サイズアレイに配列するためのプログラム命令と、
前記非零変換係数の数量が前記固定サイズアレイの容量を超えているかどうかを判定するためのプログラム命令と、
前記固定サイズアレイの容量を超える非零変換係数を可変サイズオーバーフローベクトルに保持するためのプログラム命令と、
前記非零変換係数を圧縮領域から空間領域に平行移動させる（translate）するためのプログラム命令とを含むことを特徴とするコンピュータ可読メディア。
【請求項１３】　前記固定サイズアレイは、複数の固定サイズブロックを含むことを特徴とする請求項１２に記載のコンピュータ可読メディア。
【請求項１４】　前記固定サイズブロックは各々、８つの非零変換係数を保持するように構成されていることを特徴とする請求項１３に記載のコンピュータ可読メディア。
【請求項１５】　データフレームのブロック毎に、前記固定サイズアレイの係数を、可変サイズオーバーフローベクトルの対応する係数に写像するためのプログラム命令をさらに含むことを特徴とする請求項１２に記載のコンピュータ可読メディア。
【請求項１６】　ハイブリッド因数分解及び整数近似技法を用いて前記非零変換係数に逆動き補償を実行するためのプログラム命令をさらに含むことを特徴とする請求

項１２に記載のコンピュータ可読メディア。
【請求項１７】　回路であって、
ビデオデコーダ集積回路チップからなり、当該ビデオデコーダ集積回路チップは、
ビデオデータのフレームと関連付けられるデータのビットストリームを受け取るための回路構成（circuitry）と、
前記データのビットストリームを変換領域表現に復号化するための回路構成と、
前記ビデオデコーダと関連付けられるメモリの中に前記変換領域表現の非零変換係数をハイブリッドデータ構造配列にするための回路構成と、
表示するために前記変換領域表現の非零変換係数を解凍するための回路構成とを含むことを特徴とする回路。
【請求項１８】　前記ビットストリームは、H.263ビットストリームであることを特徴とする請求項１７に記載の回路。
【請求項１９】　前記メモリは、前記ビデオデコーダ集積回路チップとは別になっていることを特徴とする請求項１７に記載の回路。
【請求項２０】　ハイブリッド因数分解及び整数近似技法により逆動き補償を実行するための回路構成をさらに備えることを特徴とする請求項１７に記載の回路。
【請求項２１】　前記メモリは、スタティックランダムアクセスメモリであることを特徴とする請求項１７に記載の回路。
【請求項２２】　ビデオ画像を表示するように構成された機器であって、
中央処理機構（CPU）と、
ランダムアクセスメモリ（RAM）と、
画像を表示するように構成されたディスプレイ画面と、
ビデオビットストリームを変換領域表現に変換するように構成されたデコーダ回路構成とからなり、当該デコーダ回路構成は、変換領域表現の非零変換係数を、デコーダ回路構成と関連付けられたメモリの中にハイブリッドデータ構造配列にする能力を有し、前記デコーダ回路構成は逆動き補償時にハイブリッド因数分解／整数近似技法を選択的に適用するための回路構成を含み、さらに、前記CPU、前記RAM、前記ディスプレイ画面、及び前記デコーダ回路構成と通信しているバスとからなることを特徴とする機器。
【請求項２３】　前記機器は、携帯用電子機器であることを特徴とする請求項２２に記載の機器。
【請求項２４】　前記携帯用電子機器は、パーソナルディジタルアシスタント、セルラー電話、ウェブタブレット、及びポケットパソコンから構成されるグループから選択されることを特徴とする請求項２３に記載の機器。
【請求項２５】　前記ハイブリッドデータ構造は、複数の固定サイズブロックを有する固定サイズアレイと、可変サイズオーバーフローベクトルとを含むことを特徴とする請求項２２に記載の機器。
【請求項２６】　前記複数の固定サイズブロックは各々、８つの非零変換係数を保持するように構成されていることを特徴とする請求項２５に記載の機器。
【請求項２７】　８を超える非零変換係数は、前記可変サイズオーバーフローベクトルの中に保持されることを特徴とする請求項２６に記載の機器。
【請求項２８】　前記デコーダ回路構成は、前記ハイブリッドデータ構造と関連付けられるデータを保持するように構成されたオンチップメモリを含むことを特徴とする請求項２２に記載の機器。
【請求項２９】　前記逆動き補償時にハイブリッド因数分解／整数近似技法を選択的に適用するための回路構成は、
ビデオ画像のフレームのブロックを、活動中の動き及び非活動中の動きのうちの一つと関連付けられると、識別するための回路構成と、
活動中の動きエリアと関連付けられたブロックには因数分解技法を適用する一方、非活動中の動きエリアと関連付けれらたブロックには整数近似技法を適用することによって、逆動き補償を実行するための回路構成とを含むことを特徴とする請求項２２に記載の機器。
【請求項３０】　前記ビデオビットストリームは、低レートビデオビットストリームであることを特徴とする請求項２２に記載の機器。
【請求項３１】　逆メモリ補償を実行するための方法であって、
ビデオビットストリームを受け取り、
半画素行列及び完全画素行列から構成されるグループから選択された変換行列タイプを識別し、
前記変換行列タイプが半画素行列ならば、
前記半画素行列に対応するビットストリームを復号化するために因数分解技法を適用し、
前記変換行列タイプが完全画素行列ならば、
前記完全画素行列に対応するビットストリームを復号化するために整数近似技法を適用することからなることを特徴とする方法。
【請求項３２】　前記ビデオビットストリームは、低レートビデオビットストリームであることを特徴とする請求項３１に記載の方法。
【請求項３３】　前記半画素行列に対応するビットストリームを復号化するために因数分解技法を適用するオペレーションは、
前記半画素行列を疎行列の列に因数分解することを含み、当該疎行列は順序行列及び対角行列を含むことを特徴とする請求項３１に記載の方法。
【請求項３４】　前記完全画素行列に対応するビットストリームを復号化するために整数近似技法を適用するオペレーションは、
前記完全画素行列の各要素を２進数と近似することを含

むことを特徴とする請求項３１に記載の方法。
【請求項３５】　各要素は、一番近い２乗に丸められることを特徴とする請求項３４に記載の方法。
【請求項３６】　ビデオデータを復号化するための方法であって、
圧縮ビットストリーム内のビデオデータのフレームを受け取り、
前記フレームのブロックを圧縮領域で変換領域表現に復号化し、
前記変換領域表現と関連付けられるデータをハイブリッドデータ構造に保持し、
前記圧縮領域で前記変換領域表現と関連付けられるデータに逆動き補償を実行することからなり、当該逆動き補償を実行することは、
前記ビデオデータフレームの一部分と関連付けられる変換行列のタイプを判定し、
逆動き補償を向上させるためにハイブリッド因数分解及び整数近似技法を適用することを含むことを特徴とする方法。
【請求項３７】　前記圧縮ビットストリームは、H.263、H.261、Motion Picture Expert Groupから構成されるグループから選択された規格と関連付けられることを特徴とする請求項３６に記載の方法。
【請求項３８】　前記ハイブリッドデータ構造は、固定サイズアレイと、可変サイズオーバーフローベクトルとを含むことを特徴とする請求項３６に記載の方法。
【請求項３９】　前記変換行列のタイプは、半画素行列及び完全画素行列からなるグループから選択されることを特徴とする請求項３６に記載の方法。
【請求項４０】　前記半画素行列は画像の高度の動き領域と関連付けられる一方、前記完全画素行列は画像の最小動き領域と関連付けられることを特徴とする請求項３９に記載の方法。
【請求項４１】　前記逆動き補償を向上させるためにハイブリッド因数分解及び整数近似技法を適用するオペレーションは、
フレームの高動き領域に対応するブロックと関連付けられる行列に因数分解技法を適用し、
フレームの残りのブロックに整数近似技法を適用することを含むことを特徴とする請求項３６に記載の方法。
【請求項４２】　前記圧縮ビットストリームは低レートビットストリームであることを特徴とする請求項３６に記載の方法。
【請求項４３】　圧縮領域で逆動き補償を実行するためのプログラム命令を有するコンピュータ可読メディアであって、
変換行列を識別するためのプログラム命令と、
前記変換行列が半画素行列及び完全画素行列のうちの一つであるかどうかを判定するためのプログラム命令と、
半画素行列に対応するビットストリームのブロックを復号化する因数分解技法を適用するためのプログラム命令と、
完全画素行列に対応するビットストリームのブロックを復号化する整数近似技法を適用するためのプログラム命令とからなることを特徴とするコンピュータ可読メディア。
【請求項４４】　前記逆動き補償を実行するためのプログラム命令は、圧縮領域で実行されることを特徴とする請求項４３に記載のコンピュータ可読メディア。
【請求項４５】　動きベクトルデータを抽出するためのプログラム命令をさらに含み、当該動きベクトルデータは変換行列を半画素行列及び完全画素行列のうちの一つとして識別することを特徴とする請求項４３に記載のコンピュータ可読メディア。
【請求項４６】　符号化されたデータフレームブロックと関連付けられる非零変換係数をハイブリッドデータ構造に配列するためのプログラム命令をさらに含むことを特徴とする請求項４３に記載のコンピュータ可読メディア。
【請求項４７】　前記完全画素行列に対応するビットストリームのブロックを復号化する整数近似技法を適用するためのプログラム命令は、
完全画素行列の各要素を２進数と近似するためのプログラム命令を含むことを特徴とする請求項４３に記載のコンピュータ可読メディア。
【請求項４８】　前記半画素行列に対応するビットストリームのブロックを復号化する因数分解技法を適用するためのプログラム命令は、
半画素行列を疎行列の列に因数分解するためのプログラム命令を含み、当該疎行列は順序行列及び対角行列を含むことを特徴とする請求項４３に記載のコンピュータ可読メディア。
【請求項４９】　回路であって、
ビデオデータを復号化するように構成された集積回路チップからなり、当該集積回路チップは、
ビデオデータのフレームと関連付けられるデータのビットストリームを受け取るための回路構成と、
前記データのビットストリームを変換領域表現に復号化するための回路構成と、
変換行列のタイプを識別するための回路構成と、
ハイブリッド因数分解及び整数近似技法により逆動き補償を実行するための回路構成とを含むことを特徴とする回路。
【請求項５０】　前記集積回路チップはさらに、
前記変換領域表現の非零変換係数をハイブリッドデータ構造に配列するための回路構成を有することを特徴とする請求項４９に記載の回路。
【請求項５１】　前記ビットストリームは低レートビットストリームであることを特徴とする請求項４９に記載の回路。

【請求項５２】　前記ハイブリッド因数分解及び整数近似技法により逆動き補償を実行するための回路構成は、半画素変換行列には因数分解技法を適用し、完全画素変換行列には整数近似技法を適用するように構成されていることを特徴とする請求項４９に記載の回路。
【請求項５３】　さらに、前記集積回路チップと通信しているメモリを含むことを特徴とする請求項４９に記載の回路。
【請求項５４】　前記因数分解及び整数近似技法はデータに対して圧縮領域で用いられることを特徴とする請求項４９に記載の回路。
【請求項５５】　ビデオデコーダであって、
入ってくるビットストリームから係数値及び動きベクトルデータを抽出するように構成された可変長デコーダ（VLD）を有し、
前記可変長デコーダと通信している反量子化ブロックを有し、当該反量子化ブロックは前記係数値をスケーリングし直すように構成されており、
前記反量子化ブロックと通信している下流のブランチを有し、当該下流ブランチは誤差係数を空間領域に復号化するように構成されており、
前記反量子化ブロックと通信している上流ブランチを有し、当該上位ブランチは内部変換領域表現を維持するように構成され、前記上流ブランチは、現ブロックを再構築するために、前記復号化された誤差係数に加算されることができる空間領域出力を生成するように構成されていることを特徴とするビデオデコーダ。
【請求項５６】　前記ビデオデコーダはソフトウェアとして実行されていることを特徴とする請求項５５に記載のビデオデコーダ。
【請求項５７】　前記ビデオデコーダはハードウェアとして実行されていることを特徴とする請求項５５に記載のビデオデコーダ。
【請求項５８】　前記入ってくるビットストリームは低レートビットストリームであることを特徴とする請求項５５に記載のビデオデコーダ。
【請求項５９】　前記上流ブランチはフィードバックループを含み、当該フィードバックループは、フレームバッファ、動き補償ブロック、離散コサイン変換ブロックを含むことを特徴とする請求項５５に記載のビデオデコーダ。
【請求項６０】　前記下流ブランチは、ランレングス復号ブロック及び逆補償ブロックを含むことを特徴とする請求項５５に記載のビデオデコーダ。
【請求項６１】　逆動き補償のオペレーションは圧縮領域で実行されることを特徴とする請求項５５に記載のビデオデコーダ。
【請求項６２】　前記変換領域表現の非零係数は、メモリ所要量を減らすために、ビデオデコーダと関連付けられるメモリの中にハイブリッドデータ構造で配列されることを特徴とする請求項５５に記載のビデオデコーダ。
【請求項６３】　前記ハイブリッドデータ構造は、固定サイズアレイ及び可変サイズオーバーフローベクトルを含むことを特徴とする請求項６２に記載のビデオデコーダ。
【請求項６４】　前記逆動き補償は、ハイブリッド因数分解及び整数近似技法を含むことを特徴とする請求項６１に記載のビデオデコーダ。
【請求項６５】　前記ハイブリッド因数分解及び整数近似技法は、半画素変換行列には因数分解技法を適用し、完全画素変換行列には整数近似技法を適用するように構成されていることを特徴とする請求項６４に記載のビデオデコーダ。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、一般的には、ディジタルビデオ技術に関し、より具体的には、効率的なメモリ圧縮法を実現するための方法及び装置に関すると共に、圧縮領域ビデオデコーダのための効率的な逆動き補償法を実現するための方法及び装置に関する。
【０００２】
【従来の技術】セルラー電話やパーソナルディジタルアシスタントといったモバイル端末でのビデオアクセスでは、モバイルシステムの性質上の限界があるために、数多くの難しい課題に出くわす。例えば、低消費電力型のハンドヘルド機器は、バンド幅、電力、メモリ、及びコスト上の必要条件により制約を受ける。こうしたハンドヘルド機器で受信されるビデオデータはビデオデコーダにより復号化される。そうした端末と関連付けられるビデオデコーダは、空間領域、つまり、解凍領域で動き補償を実行する。H.263、H261、MPEG1/2/4といったビデオ圧縮規格は、低ビットレートでビデオを符号化するのに動き補償型離散コサイン変換（DCT）スキームを用いる。ここで採用している低ビットレートとは、秒毎約６４キロビット未満のビットレートのことである。DCTスキームでは、時間的な冗長度を削除するために動き予測（ME）及び動き補償（MC）を用いる一方、それ以外の空間的冗長度を削除するためにDCTを用いる。
【０００３】図１は、ビデオデータを復号化すると共に、空間領域で動き補償を実行するためのビデオデコーダの概略図である。ビットストリーム１０２がデコーダ１００によって受け取られる。デコーダ１００には、可変デコーダ（VLD）ステージ１０４と、ランレングスデコーダ（RLD）ステージ１０６、反量子化（DQ）ステージ１０８、逆離散コサイン変換（IDCT）ステージ１１０、動き補償（MC）ステージ１１２、及びフレームバッファとも呼ばれるメモリ（MEM）１１４とがある。最初の４ステージ（VLD１０４、RLD１０６、DQ１０８、IDCT１１０）は、圧縮されているビットストリームを復号化して画素領域に戻す。イントラコード化された（intrac

oded) ブロックでは、現フレームの中でブロックを再構築するために最初の４ステージ、つまり、104、106、108、110の出力が直接用いられる。インターコード化された (intercoded) ブロックでは、出力が予測誤差を表わしており、現フレームの中でブロックを再構築するためにその前のフレームから作られた予測に出力が付加される。よって、現フレームはブロック単位で再構築される。最終的に、現フレームがデコーダ、つまり、ディスプレイ１１６の出力に送られると共に、フレームバッファ (MEM) １１４にも保持される。

【０００４】MEM１１４は、動き補償１１２に必要な既に復号化されている画像 (picture) を保持している。MEM１１４のサイズは、入ってくる画像フォーマットに応じてスケーリングしなければならない。例えば、H.263は5つの規格化された画像フォーマット、すなわち、(1) 1/4以下 (sub-quarter) 共通中間フォーマット (サブQCIF)、(２) 1/4共通中間フォーマット (QCIF)、(３) 共通中間フォーマット (CIF)、(４) 4CIF、及び (５) 16CIF、をサポートしている。各フォーマットは、画像の幅及び高さだけでなく、アスペクト比も定義する。広く知られているように、画像は1つの輝度成分及び2つの色差成分 (Y, Cr, Cb) としてコード化される。それらの成分は4:2:0の構成でサンプリングされ、各成分は画素あたり8ビット分解能を有する。例えば、図1のビデオデコーダは、CIFフォーマットのH.263ビットストリームを復号化しながら、MEM１１４のために約２００キロバイトのメモリを割り当てなければならない。さらに、ビデオ会議システムで欠かせないように、複数のビットストリームが一度に復号化される場合、メモリ需要が大きくなりすぎる。

【０００５】MEM１１４は、ビデオデコーダ１００の中で唯一最大のメモリ使用源である。メモリ使用を軽減するために、入ってくるビットストリームの色成分の分解能を低下させることが一つのアプローチとして考えられる。例えば、モバイル端末上のカラー表示の濃度が65,536色しか表示できないとしたら、色成分 (Y, Cr, Cb) の分解能を画素につき24ビットから16ビットに低下させることができる。この技法は、可能性としては、メモリ使用度を30%減らすことができるけれども、ビデオデコーダで回路的に対応しなければならないディスプレイ依存型ソリューションである。また、この技法は、ピーク信号対雑音比 (PSNR) 要件を変えて簡単にスケーリングすることができないので、自由度がない。

【０００６】空間領域でデータに操作を行なうには、圧縮領域処理と比べ、より大きなメモリ容量が要る。空間領域では、動き補償を算出すると共に連続フレームの画像に動き補償をかけることが容易である。しかしながら、圧縮領域で操作している場合、動き補償は、誤差値がもはや空間値ではなくなるから、つまり、圧縮領域で操作している時の誤差値は画素値ではないから、動きベクトルが前のフレームを示すのと比べてそれほど明快ではない。その上に、圧縮領域データを効率的に処理する能力を有する方法がない。先行技術のアプローチは主に、圧縮領域をトランスコード化し、スケーリングし、鮮鋭化する各アプリケーションを中心にしている。さらに、圧縮領域対応の逆補償用アプリケーションは、ピーク信号対雑音比 (PSNR) の性能が貧弱になる傾向があると同時に、１秒あたりに表示可能なフレーム量の点から、応答時間が受容れ難いほど遅い。

【０００７】

【特許文献１】米国再発行特許発明第6,240,210号明細書

【特許文献２】米国特許第6,157,740号明細書

【０００８】

【発明が解決しようとする課題】そこで、低ビットレートのビデオデータを復号化するのに要するメモリ所要量を最小限にする方法並びに装置を提供すると共に、圧縮領域ビデオレコーダの高速且つ効率的な逆動き補償を可能にする方法並びに装置を提供するべく、先行技術が抱える問題を解決する必要がある。

【０００９】

【課題を解決するための手段】おおまかに言えば、本発明は、ハイブリッドデータ構造を採用することによりメモリ所要量を最小限にするように構成されたビデオデコーダを提供することで、こうしたニーズの少なくとも一つの局面を満たすものである。なお、本発明のこの態様は、方法、システム、コンピュータ可読媒体、又はデバイスなど、いろいろな方法で実現することができる。本発明のこの態様の実施例を以下にいくつか説明する。

【００１０】１実施例において、ビットストリームを復号化するのに要するメモリ所要量を低減するための方法を提供する。この方法は、ビデオビットストリームを受け取ることから始まる。次に、そのビットストリームのフレームが変換 (例えば、離散コサイン変換 (DCT)) 領域表現に復号化される。次に、その変換領域表現の非零係数が識別される。次に、ハイブリッドデータ構造がアセンブルされる。このハイブリッドデータ構造は、固定サイズのアレイ及び可変サイズのオーバーフローベクトルを含んでいる。次に、変換領域表現の非零係数がハイブリッドデータ構造の中に挿入される。

【００１１】別の実施例において、ビデオデータを復号化するための方法を提供する。この方法は、圧縮されたビットストリーム内のビデオデータのフレームを受け取ることから始まる。次に、そのフレームのブロックが、圧縮領域で変換 (例えば、DCT) 領域表現に復号化される。次に、ハイブリッドデータ構造が定義される。次に、その変換領域表現と関連付けられるデータがハイブリッドデータ構造で保持される。次に、圧縮領域で変換領域表現と関連付けられるデータに対して逆動き補償が実行される。データに逆動き補償を行なった後、表示す

るためにデータが解凍される。

【００１２】また別の実施例において、低レートのビットストリームデータをハイブリッドデータ構造で保持するべく並べ替えるためのプログラム命令を有するコンピュータ可読媒体を提供する。このコンピュータ可読媒体には、データフレームの符号化されたブロックと関連付けられる非零変換（例えば、DCT）係数を識別するためのプログラム命令が入っている。その非零変換係数を固定サイズアレイに配列するためのプログラム命令が入っている。非零変換係数の数量が固定サイズアレイの容量を超えたかどうか判定するためのプログラム命令を提供する。固定サイズアレイの容量を超える非零変換係数を可変サイズオーバーフローベクトルで保持するためのプログラム命令、及び非零変換係数を圧縮領域から空間領域に平行移動させる（translate）ためのプログラム命令を含んでいる。

【００１３】さらに別の実施例において、回路を提供する。この回路は、ビデオデコーダ集積回路チップを有する。このビデオデコーダ集積回路チップは、ビデオデータのフレームと関連付けられるデータのビットストリームを受け取るための回路構成を含む。ビデオデコーダには、データのビットストリームを変換（例えば、DCT）領域表現に復号化するための回路構成が入っている。変換領域表現の非零変換係数を、ビデオデコーダと関連付けられるメモリの中のハイブリッドデータ構造で配列するための回路構成を提供する。表示するために、変換領域表現の非零変換係数を解凍するための回路構成も提供する。

【００１４】別の実施例において、画像を表示するように構成された機器を提供する。この機器は、中央処理機構（CPU）、ランダムアクセスメモリ（RAM）、及び画像を表示するように構成されたディスプレイ画面とを含む。ビデオビットストリームを変換（例えば、DCT）領域表現に変換するように構成されたデコーダ回路構成を含む。このデコーダ回路は、変換領域表現の非零変換係数を、デコーダ回路と関連付けられるメモリの中にハイブリッドデータ構造で配列する能力を有する。デコーダ回路には、逆動き補償時に、ハイブリッド因数分解／整数近似の技法を選択的に適用するための回路構成が入っている。CPU、RAM、表示画面、及びデコーダ回路と通信しているバスも有する。

【００１５】大まかに言えば、本発明は、メモリ所要量を低減すると同時に一応満足できるビデオ画質を提供し、それと同時に、圧縮領域で逆動き補償を実行する能力を有するビデオデコーダを提供することにより、こうしたニーズの少なくとも別の局面を満たすものである。なお、本発明のこの態様は、方法、システム、コンピュータ可読媒体、又は機器など、いろいろな方法で実現することができる。本発明のこの態様の実施例についていくつか以下に説明する。

【００１６】一つの実施例において、逆メモリ補償を実行するための方法を提供する。この方法は、ビデオビットストリームを受け取ることから始まる。次に、変換行列タイプが識別される。この変換行列タイプは、半画素行列か完全画素行列かのどちらかである。この方法は、変換行列タイプが半画素行列ならば、その半画素行列に対応するビットストリームを復号化する因数分解技法を適用することを含む。変換行列タイプが完全画素行列ならば、その完全画素行列に対応するビットストリームを復号化する整数近似技法を適用することを含む。

【００１７】別の実施例において、ビデオデータを復号化するための方法を提供する。この方法は、圧縮されたビットストリーム内のビデオデータのフレームを受け取ることから始まる。次に、そのフレームのブロックが変換（例えば、離散コサイン変換（DCT））領域表現に圧縮領域で復号化される。次に、その変換領域表現と関連付けられるデータがハイブリッドデータ構造で保持される。次に、その圧縮領域で変換領域表現と関連付けられるデータに逆動き補償が実行される。逆動き補償の実行には、ビデオデータのフレームの一部と関連付けられる変換行列のタイプを決めることと、逆動き補償を向上させるためにハイブリッド因数分解及び整数近似技法を適用することが含まれる。

【００１８】また別の実施例において、圧縮領域で逆動き補償を実行するためのプログラム命令を有するコンピュータ可読メディアを提供する。このコンピュータ可読メディアには変換行列を識別するためのプログラム命令が入っている。変換行列が半画素行列か或いは完全画素行列かを判定するためのプログラム命令が入っている。半画素行列に対応するビットストリームのブロックを復号化する因数分解技法を適用するためのプログラム命令並びに完全画素行列に対応するビットストリームのブロックを復号化する整数近似技法を適用するためのプログラム命令を含んでいる。

【００１９】さらに別の実施例において、回路を提供する。この回路には、ビデオデータを復号化するように構成された集積回路チップがある。この集積回路チップは、ビデオデータのフレームと関連付けられるデータのビットストリームを受け取るための回路構成を含んでいる。集積回路チップには、データのビットストリームを変換（例えば、DCT）領域表現に復号化するための回路構成が搭載されている。変換行列のタイプを識別するための回路構成並びにハイブリッド因数分解及び整数近似技法によって逆動き補償を実行するための回路構成が集積回路チップに搭載されている。

【００２０】別の実施例において、ビデオデコーダを提供する。このビデオデコーダは、入ってくるビットストリームから係数値及び動きベクトルデータを抽出するように構成された可変長デコーダ（VLD）を含んでいる。VLDと通信している反量子化ブロックを有する。この反量

子化ブロックは、係数値をスケーリングし直すように構成されている。その反量子化ブロックと通信している下流のブランチが設けられている。この下流ブランチは誤差係数を空間領域に復号化するように構成されている。反量子化ブロックと通信している上流のブランチを含んでいる。この上流ブランチは、内部変換（例えば、DCT）領域変換を維持するように構成されている。上流ブランチはさらに、現ブロックを再構築するために復号化された誤差係数に加算されることが可能な空間領域出力を生成するように構成されている。

【００２１】本発明のその他の態様並びに効果は、発明の原理を例を挙げて示した添付の図面と共に、以下に述べる詳細な説明から明白になる。

【００２２】

【発明の実施の形態】本発明を、圧縮領域ビデオ復号化に要するメモリ容量を最小限にするためのシステム、装置、及び方法として説明する。しかしながら、当業者ならば、以下の説明に鑑みて、以下に説明する詳細を部分的に又は全く知らなくても本発明を実施できることが分かる。また、本発明を不要に不明瞭なものにしないために、既によく知られているプロセスオペレーションについては詳細に説明しない。図１については、「従来の技術」の項で説明した。本明細書で使用している「約」という言葉は、基準値の＋／－１０％のことである。

【００２３】ここで説明する実施例は、圧縮領域でビデオデータを復号化する際に使用されるメモリの低減を可能にするデータ構造を提供する。１実施例では、周波数領域、つまり、圧縮領域で、現フレームが保持され、逆動き補償が実行されるように、ビデオ復号化パイプラインが並べ替えられている。ハイブリッドデータ構造は、計算コストやデータの有意な損失なしに、圧縮領域でのデータの操作を可能にする。１実施例において、ハイブリッドデータ構造は符号化されたブロックの中に非零離散コサイン変換（DCT）係数はほんのわずかしかないという事実を利用している。従って、フレーム全体の非零DCT係数だけが保持されるので、メモリ所要量を低減することができる。以下により詳細に説明するように、ハイブリッドデータ構造は固定サイズのアレイと可変サイズのオーバーフローベクトルとを含んでいる。可変サイズオーバーフローベクトルは、固定サイズアレイの容量を超える符号化されたブロックの非零DCT係数を保持する。

【００２４】図２は、本発明の１実施例による、逆動き補償が実行されるように配置構成されたビデオデコーダの概略図である。ここで、ビデオデコーダ１２０によってビットストリーム１２２が受け取られる。最初の２ステージ、つまり、可変長デコーダ（VLD）ステージ１２４及び反量子化（DQ）ステージ１２６は圧縮されたビットストリームをDCT領域表現に復号化する。DCT領域表現は、動き補償（MC）ステージ１３４で使用するために、フレームバッファとも呼ばれるメモリ（MEM）１３０に保持される。MC１２８及びMEM１３４を含んだ動き補償フィードバックループの後に、ランレングスデコーダ（RLD）ステージ１３２及び逆DCT（IDCT）ステージ１３４が実行される。従って、復号化されたブロックの内部表現は圧縮領域のままである。符号化されたブロック内に非零DCT係数はほんの小数しかないので、この特徴を、フレーム内の各ブロックの非零DCT係数しか保持しないMEM１３０のデータ構造を開発することによって、利用することができる。以下により詳細に明らかにしているように、ハイブリッドデータ構造により可能になるメモリ圧縮は、ビデオ画質の損失なしにメモリ使用を５０％減らすことができる。人間の視覚系は、高位のDCT係数よりも低位DCT係数に対して敏感だから、以下に説明するように、高位DCTをフィルタ処理して取り除くと共にメモリ使用対変動電力又はピークの信号対雑音比の要件をトレードオフするしきい値化スキームを開発した。

【００２５】そこで、高速でしかもメモリ効率のいい復号化ができるように最適化される完全圧縮領域ビデオ復号化について説明する。１実施例で、本書で言及しているテストのために、パブリックドメインH.263に準拠しているデコーダであるTELENORのビデオデコーダを使用した。なお、以下に説明する実施例の中にはH.263ビットストリームと呼んでいるものがあるが、実施例はH.263ビットストリームに対する操作だけに限らない。すなわち、Motion Picture Expert Group（MPEG）1/2/4、H.261など、ビデオデータを有するどんなDCTベースの圧縮ビットストリームでも採用することができる。圧縮領域での効率的な処理を可能にする離散コサイン変換（DCT）領域表現のための高速逆動き補償アルゴリズムは数多い。なお、符号化されたブロック内に非零DCT係数を保持するメモリ圧縮法は、圧縮領域での処理だから、メモリ所要量を低減することが可能になる。さらに、スピード及びメモリの最適化における様々な性能のトレードオフを実証するために、本書で説明している逆動き補償技術及びメモリ圧縮を用いた圧縮領域処理を採用するビデオデコーダの性能を３つの次元で、つまり、計算量、メモリ効率、PSNRの観点から評価する。

【００２６】

図３は、空間領域で実行される逆動き補償を説明している概略図である。ここでは、基準フレームの中の動き補償がなされたブロックに基づいて現ブロックの予測が行なわれる。現フレーム１４４の現在の８ｘ８の空間ブロック f_k １４２が、基準フレーム１４６の中の４つの基準ブロック f_1', f_2', f_3' 及び f_4' １４４－１から１４４－４から導出される。基準ブロックは、f_kの変位を動きベクトル（Δx,Δy）によって算出し、基準フレームの中で動きベクトルが交差するブロックを選ぶことにより選択される。（Δx>0, Δy>0）の場合、f_kは右下に転置される。f_kのf_1'とのオーバラップから、オーバーラップパラメータ（w, h）並びに隣接ブロックとのパラメータ（8−w, h），（w, 8−h）及び（8−w, 8−h）を判定することができる。

【数１】

【００２７】

$$f_k = \sum_{i=1}^{4} c_{i1} f_i' c_{i2} \qquad (2)$$

各ブロックを８ｘ８の行列として表現することができるから、行列の再構築を、窓かけ及びシフトされた行列 $f_1', \ldots, f_4'$の総和として記述することができる。方程式（２）で、行列 c_{ij}, i = 1,...,4, j = 1,2は、f_i'に対して窓掛け及びシフト操作を実行する。行列 c_{ij}は、１と０（ゼロ）からなる８ｘ８の疎行列である。また、c_{ij}はオーバーラップパラメータ(w, h)の関数で、次のように定義される：

【数２】

【００２８】

$$c_{11} = c_{21} = U_h = \begin{pmatrix} 0 & I_h \\ 0 & 0 \end{pmatrix}, \qquad (3)$$

$$c_{12} = c_{32} = L_w = \begin{pmatrix} 0 & 0 \\ I_w & 0 \end{pmatrix}, \qquad (4)$$

ここで、I_h及び I_wは次元 $h \times h$ 及び $w \times w$ それぞれの恒等行列である。同様に、

【数３】

$$c_{31} = c_{41} = L_{8-h}, \qquad (5)$$

$$c_{22} = c_{42} = L_{8-w}. \qquad (6)$$

【００２９】

DCT領域における逆動き補償により、動き補償がなされたインターコード化されたブロックからイントラコード化されたブロックが再構築される。このコンセプトは空間領域に類似している。但し、例外は、全ての係数がDCT領域に保たれる、つまり、$f_1', \ldots, f_4'$のDCT、つまり、$F_1', \ldots, F_4'$から、f_kのDCT、つまり、F_kが直接再構築される点である。

Sは、２次元DCTの８ｘ８の基底ベクトルを含む行列として定義される。DCT変換$S^tS = I$の単位的性質を用いて、方程式（２）が、

【数４】

$$f_k = \sum_{i=1}^{4} c_{i1} S^t S f_i' S^t S c_{i2}. \qquad (7)$$

と等価であることを実証することができる。

方程式（７）の両辺に S を前に乗じて（premultiply）、S^t を後に乗じる（postmultiply）と、

【数５】

$$F_k = \sum_{i=1}^{4} C_{i1} F'_i C_{i2}, \qquad (8)$$

になる。

ここで、C_{ij}は、c_{ij}のDCTである。方程式（８）は、前後に乗じられた項$F'_1,\ldots,F'_4$の総和として、F_kを算出する。行列 C_{ij}は、変換列、つまり、逆 DCT、窓掛け、シフト、順方向DCTを含む単一の複合行列である。従って、方程式（８）は、行列乗算だけを用いて、$F'_1,\ldots,F'_4$からF_kを直接算出する方法を記述している。これらの行列乗算は、空間領域及び周波数領域間で明白に変換しなくても、DCT領域で行なうことができる。しかしながら、説明した行列乗算は容認し難いほど遅い。その結果、毎秒約５フレームしか表示できないので、表示画質が劣る。行列乗算は容認しがたい遅延を起こすボトルネックになるので、以下に説明するDCT領域逆動き補償アルゴリズムは、こうした行列乗算の計算量の低減を狙っている。

【００３０】低ビットレートビデオ、つまり、毎秒約６４キロビット未満のビットレートを有するビデオデータは、ビデオ会議のアプリケーションに使用されると共に、セルラー電話、パーソナルディジタルアシスタント（PDA）や、その他のハンドヘルド機器や電池で動く機器のワイヤレスビデオといったアプリケーション向けである。H.263規格は、低ビットレートのビデオ復号化用のビットストリームシンタックス及びアルゴリズムを指定している模式的規格である。アルゴリズムは、変換符号化、動き推定／補償、係数量子化、ランレングス符号化を含む。ベースライン指定とは別に、この規格のバージョン２は、符号化性能を向上させると共にエラー差回復力を提供する１６の交渉可能なオプションもサポートしている。

【００３１】低ビットレートで符号化されたビデオは、目に見える歪みが発生する可能性がある。特に、アクションの多い分類になっているビデオ、つまり、活動的な動きのブロックはそうである。先に触れたように、本書で説明している実施例では、H.263規格のことを指しているが、適していればどんなビデオコーデック規格でも実施例と共に使用することができる。参考までに、H.263規格の機能特徴のいくつかを以下に説明するが、これにより本発明をH.263規格と共に使用することに限定しようとしているわけではない。H.263規格の一つの特徴は、この規格の中に画像群（GOP）及び高位レイヤーが存在していない点である。ベースライン符号化された列が、単一イントラフレーム（Iフレーム）とそれに続く長いインターフレーム（Pフレーム）列とだけからなる場合、時間的冗長が連続フレーム間で取り除かれるので、長いPフレーム列がより高い圧縮比を実現する。しかしながら、動き推定／動き補償（ME/MC）は時間依存性を生じさせるので、損失性符号化プロセス時に発生した誤差が復号化プロセス時に集積する。Iフレームが足りないと、デコーダはこの誤差集積を崩すことができない。H.263規格は強制更新メカニズムを有するので、符号化プロセス時に少なくとも１３２回に一回エンコーダがマクロブロックをイントラブロックとして符号化しなければならない。図４は、強制更新メカニズムの効果を説明している図である。図４に示すように、ビデオのPSNRは無作為に変動するが、列の後半にあるフレームではどんな方向にもドリフトしない。

【００３２】図５は、H.263規格で半画素の値の判定を説明している概略図である。よく知られているように、H.263規格は動き補償に半画素補間を採用している。この規格では、半画素補間が０．５分解能（つまり、<;7.5, 4.5>;）を有する動きベクトルによって示される。エンコーダは、水平方向だけ、垂直方向だけ、或いは水平垂直両方向で、補間を指定することができる。図５に示されているように、半画素値は、半画素の位置を取り巻く整数画素位置の双線形補間によって見出される。画素位置A 150-1、画素位置B 150-2、画素位置C 150-3、画素位置D 150-4は、整数画素位置を表わしているのに対し、画素位置e 152-1、画素位置f 152-2、画素位置g 152-3は、半画素位置を表わしている。水平方向の補間をe=(A+B+1)>;>;1と表わし、垂直方向の補間をf=(A+C+1)>;>;1と表わすことができる。水平垂直両方向の補間をg=(A+B+C+D+2)>;>;2と表わすことができる。

【００３３】図６A及び６Bはそれぞれ、ベースライン空間ビデオデコーダ及び圧縮領域ビデオデコーダの概略図である。図６Bのブロック図は、図６Aの空間領域ビデオデコーダの機能ブロックを部分的に並べ替えたものである。特に、RLD１３２とIDCT１３４がMC２８フィードバ

ックループの後に移動している。この配列により、圧縮領域でビデオの内部表現を保つことができる。図６Bの配列では、圧縮領域後処理モジュールをMC１２８フィードバックループのすぐ後に挿入可能である。なお、複合化（compositing）、スケーリング、非ブロック化など、特定のビデオ操作は、空間領域での操作と比べ、圧縮領域のほうが高速である。しかしながら、ビデオコーデックの観点から言えば、空間的エンコーダは圧縮領域デコーダに完全にマッチしない。図６Bに表示されているように、圧縮領域ビデオデコーダは、復号化パイプライン沿いのいくつかの点で図６Aの空間領域ビデオデコーダとは異なる。単なるブロックの並べ替えだけというのではなく、相違点は、クリッピングや丸めなど非線形操作を表わしている。これらの非線形性を有する点が、２つの領域間でPSNR測定値が異なるビデオを生成する。

【００３４】

非線形点に、（i）、（ii）、（iii）、（iv）、（v）とラベルを付けた。図６Aの空間的デコーダで、IDCT ブロック１３４は、入ってくる８ｘ８ブロックを周波数領域から空間領域に変換する。空間領域値は、色チャネル（Y, Cr, Cb）の画素値か予測誤差値かのどちらかを表わしている。図６A の点（i）で、空間値は$(-255 \le x \le 256)$の範囲にクリッピングされる。なお、図６B の DCT 係数のこのステージには同等のクリッピング操作はない。第２の相違点は、動き補償時に発生する。図６A の MC ブロック１２８は、現在の動きベクトルが参照した MEM １３０からの画素値をリターンする。図６A の点（ii）で、もし指定されていれば、半画素（HP）補間１６０が近傍の画素値の平均をとり、その結果を最も近い正の整数に丸める。図６B の点（iv）で、半画素（HP）補間１６０が、直接、DCT係数に演算を行なって、その結果を最も近い正又は負の整数に丸める。予測誤差を予測値に加算した後、もう一つの相違点が発生する。図６A の点（iii）で、和は、ブロック１６２ｂで$(0 \le x \le 255)$の範囲にクリッピングされた画素値を表わす。なお、図６B で、同じような画素値のクリッピングは、動き補償フィードバックループから復号化パイプラインの最終ステージにブロック１６２（点 v）に移動している。

【００３５】当業者ならば、MEM１３０は、動き補償の前のフレームを保持するフレームバッファであることが分かる。空間領域デコーダでは、フレームバッファが、（Y,Cr, Cb）値を保持するために、入ってくるフレームサイズに十分対応できるメモリを割り当てる。例えば、４：２：０でサンプリングされたCIFビデオには約２００キロバイトのメモリがなければならない。MEM１３０はビデオデコーダで唯一の最大メモリ使用源だから、ここで定義しているハイブリッドデータ構造及び逆動き補償により、圧縮領域復号化パイプラインのメモリ使用の軽減が可能になる。１実施例では、復号化されたビデオで有意な画質の損失なく、２倍から３倍のメモリ圧縮を達成している。

【００３６】図７は、本発明の１実施例による、ビデオ符号化及び復号化プロセス時のブロック変換を説明しているブロック図である。点線１７０より上の変換のシーケンスは、動き補償／動き推定後に、Iフレームの中のブロック又はPフレームの中のブロックに対してビデオエンコーダが用いる空間圧縮法を説明している。画素ブロック１７２は完全８ｘ８行列である。この時点で、空間領域での圧縮又は打ち切りはどんなものでも、再構築

RL = binary rrrr111111111111'

最下位の１２ビット（111111111111'）は、ブロック１８４から反量子化されたDCT係数の値を定義している。

(9)

されたブロックの目に見える画質に直接影響する。しかしながら、DCT変換後、変換された行列１７４はコンパクトで、低周波数で項が大きくなっている。量子化のステップが、ブロック１７６の中の高周波数の小さめの項を零にすることにより、ブロックをさらにコンパクトにする。ブロック１７６で強調されているジグザグ走査が、低周波数から高周波数にDCT係数に順番を付ける。ランレングス符号化は、２値をもつ要素（element）、例えば、ラン及びレベルのコンパクトなリストの中の、零係数を無視し、非零DCT係数だけをランレングス表現１７８で表わす。従って、非零DCT係数のランレングス表現を保持及びアクセスする効率的なデータ構造及び方法を開発することにより、DCT領域でメモリ圧縮を達成することができる。

【００３７】１実施例において、半圧縮（SC）表現が、一つのそうしたメモリ効率のいいランレングス表現である。非零DCT係数のランレングス表現は、図７のランレングス表現１７８及び１８０に類似している。しかしながら、２つの変形がある。各２値をもつ要素（ラン、レベル）をその形の複合１６ビット値によって記述する。

【００３８】

ブロック１８４は、量子化されたブロック１８２から導出されたものである。なお、ブロック１８４は、DCT領域表現の一例である。当業者ならば、DCT係数の値は２０４８から２０４７の範囲であることが明らかである。図7のブロック１８６は、ブロック１８４にIDCT操作を行なった後のブロック１７２の再構築ブロックである。最上位の４ビット（rrrr'）はランの値を定義している。ランは、８ｘ８ブロック内のジグザグ走査に基づく、終わりの非零DCT係数の位置に対する非零DCT係数の相対位置を表わしている。非零係数のランが１５を超える可能性があるので、ランをより小さな単位に分かるためにエスケープシーケンスが定義される。１５の零係数の後に零振幅（zero amplitude）の係数が続くランを表わすために、エスケープシーケンスRL='F0'が定義されている。

【００３９】メモリ所要量を減らすために、SC表現を保持及びアクセスするためのデータ構造を開発しなければならない。次のデータ構造、つまり、アレイ、連結リスト、ベクトル、ハイブリッドを考察した。これらの構造を開発する際、メモリ圧縮の必要性と、計算量を低く維持する必要性とのバランスを考慮に入れた。以下に、表１を参照してさらに説明する。SC表現は狙ったメモリ圧縮を実現するが、特定のデータ構造によっては3つの分野でデコーダの計算量が大きく増大する。第1に、2バイト表現を採用することにより、（ラン、レベル）の値を直ちに使用することができなくなる。これらの値にアクセス及び変更を加えるには毎回、ビットをパック及びアンパックするための関数が必要である。第2に、コンパクトランレングス表現により動き補償が複雑になる。第3に、予測に予測誤差を加算するには分類及び併合（sort and merge）操作が必要である。

【００４０】図8は、ランレングス表現の各８ｘ８ブロックの開始位置を見出すために、別のインデックスを使用することを説明している概略図である。フレームの中の全ての８ｘ８ブロック１９２－１から１９２－４のランレングス表現を保持するためにベクトルとも呼ばれる単一リスト１９０を用いる場合、動き補償時に或る特定のDCTブロックにアクセスするにはその開始位置を調べるために別のインデックスが必要になる。それは、動き補償を複雑にしてしまう。

【００４１】図9A及び9Bは、アレイベースのデータ構造及びリストデータ構造それぞれに関して、予測に予測誤差を加算するのに必要な分類及び併合操作を説明している。図9Aで、アレイベースのデータ構造は、対応するアレイインデックスにおける値の加算しか必要としない。しかしながら、アレイベースのデータ構造にはメモリ圧縮効果がない。図9Bで、リスト（もしくは、ベクトル）データ構造にはさらに分類及び併合操作が必要である。すなわち、併合のアルゴリズムには挿入及び削除の機能がなければならない。それは、ベクトルなどデータ構造にとって計算量の観点から非常に高価になる。より具体的に言えば、インデックスが等しければ、DCT係数を加算又は除算することができる。例えば、(0,20) + (0,620)=(0,640)。間違ったインデックスが予測で先行している場合には、DCT係数が挿入される。例えば、(0,-3)を挿入。DCT値の加算により０になる場合には、DCT係数は削除される。例えば、(1,13) + (4,-13)=(1,0)。

【００４２】表１は、いろいろなデータ構造のメモリ圧縮比及び計算費用を比較したものである。アレイベースのデータ構造は、予測の更新に必要な６４の加算以外に追加の計算費用は発生しないが、DCT係数のアレイは、各DCT係数が保持のために１バイトではなく２バイトを必要とするので、画素のアレイと比べメモリ圧縮を実現することはできない。連結リスト又は半圧縮（SC）表現のベクトルは、画素のアレイと比べ、上限で２．５倍のメモリ圧縮が可能である。しかしながら、ベクトルの挿入／削除費用は高価だから、いずれの解法とも最適とはいえない。特に、ベクトルの真ん中の挿入及び削除は高価である。また、リストの要素毎に内部ポインタが作成されるので連結リストのメモリオーバーヘッドは高価である。

【表１】

データ構造	メモリサイズ（キロバイト）	挿入／削除費用	メモリオーバーヘッド	圧縮比
画素アレイ	152	なし	なし	なし
DCT アレイ	304	なし	なし	なし
SC ベクトル	60	高価	最低	2.5:1
SC の連結リスト	60+オーバーヘッド	中程度	高価	2.5:1（オーバーヘッドなし）
SC のハイブリッド	70	中程度	最低	2.2:1

【００４３】SC表現のハイブリッドデータ構造により、表1で競合している利害関係を最適なバランスをすることができる。図９Aのアレイ構造の低計算費用及び図９Bのベクトル構造の高圧縮比をうまく利用するために、ハイブリッドデータ構造が開発された。ハイブリッドデータ構造は、ブロック毎に固定数のDCT係数を保持する固定サイズアレイと、これらのブロックで固定サイズアレイ割り当てを超えたDCT係数を保持する可変サイズのオーバーフローベクトルとで構成されている。なお、固定サイズアレイは、ブロック毎に適した任意数のDCT係数

を保持するように構成することができる。ここで、DCT係数の数は６４未満である。言うまでもなく、固定サイズアレイが大きくなると、それに比例してメモリ圧縮量が減る。１実施例で、固定サイズアレイは、ブロックあたり８つのDCT係数を保持するように構成されている。

【００４４】図１０は、本発明の１実施例による、メモリ圧縮及び計算効率性を可能にするアレイ構造及びベクトル構造を含むハイブリッドデータ構造の概略図である。DCTブロック２００－１、２００－２、２００－ｎには、零DCT係数及び非零DCT係数が入っている。なお、DCTブロック２００－１から２００－ｎは、図２において先に説明したようにDCT領域表現を表わす。さらに、ブロック２００－１から２００－ｎは、ビデオデータのフレームのブロック、例えば、図7のブロック１８４、と関連付けられる。ブロック２００－１から２００－ｎの各ブロックの非零係数が識別され、固定サイズアレイ２０２データ構造の中に挿入される。固定サイズアレイ２０２には、固定サイズブロック２０４－１から２０４－ｎがある。１実施例で、各ブロック２０４－１から２０４－ｎは、８ｘ８データ構造の中に８つのDCT係数を保持できる大きさになっている。なお、本発明は、８つのDCT係数を保持する構成になったブロックに限定されるものではなく、適した任意のサイズを採用することができる。先に述べたように、ブロックの容量が増大すると、それに比例してメモリ圧縮量は低下する。

【００４５】引き続き、図１０において、DCTブロック２００－１から２００－ｎのどれかに８個を超える非零係数が入っている場合、それぞれ固定サイズブロック２０４－１から２０４－ｎの容量を超える非零DCT係数はオーバーフローベクトル２０６の中に入れられる。オーバーフローベクトル２０６は、可変サイズオーバーフローベクトルとして構成されている。つまり、オーバーフローベクトルは動的である。例えば、ブロック２００－１は、９つの非零DCT係数A１～A９を含んでいる。ここで、DCT係数A１～A８は固定サイズブロック２０４－１にコピーされるが、DCT係数A９はオーバーフローベクトル２０６にコピーされる。ブロック２００－２は、１０個の非零DCT係数B１～B１０を含んでいる。ここで、DCT係数B１～B８は固定サイズブロック２０４－２にコピーされるが、DCT係数B９及びB１０は、フレームのブロック毎に、オーバーフローベクトル２０６、その他に、コピーされる。インデックス表２０８には、オーバーフローベクトル２０６のエントリに関して、対応する固定サイズブロック２０４－１から２０４－ｎを識別するエントリが入っている。各エントリは１バイトだから、インデックス表のサイズは無視することができる。従って、DCTブロック２００－１から２００－ｎに対応するデータのフレームでは、イメージ２１０を生み出すために、固定サイズアレイ２０２及びオーバーフローベクトル２０６からのデータが組み合わされる。相当なメモリの節約ができる。すなわち、DCTブロック２００－１から２００－ｎは、ほとんどの場合、６４個の零及び非零係数から、固定サイズブロック２０４－１から２０４－ｎに保持された８個以下の非零係数まで減少する。言うまでもなく、多少の非零係数が提供される可能性はあり、非零係数が８を超えるとオーバーフローベクトル２０６に保持される。

【００４６】図１１A～１１Cは、本発明の１実施例による、ハイブリッドデータ構造の固定サイズアレイの固定サイズブロックの容量及びオーバーフローベクトルを判定する際に評価される因子を説明しているグラフである。図１１Aで、２つの典型的なCIFシーケンスの輝度ブロックあたりの非零DCT係数の平均数が、線２２０及び２２２で描かれている。ブロックあたりの非零係数の数は3から７の範囲である。すなわち、６４個の係数のうち、平均して２から７個の係数だけが非零係数である。図１１Bは、固定サイズアレイが増加すると、オーバーフローベクトルのサイズが減少するので、ベクトルの挿入及び削除費用を最小限にすることができることを、図１１Aの情報を指針として用いて、説明している。ここで、線２２０－１は図１１Aの線２２０のCIFシーケンスに対応しているのに対し、線２２２－１は図１１Aの線２２２のCIFシーケンスに対応している。当業者ならば、固定サイズアレイは容量が増えるにつれて、メモリ圧縮は低下することが分かる。さらに、図１１Cは、アレイの負荷因子（load factor）も減少して、アレイはほとんど空のままであることを示している。１実施例では、ブロックあたり８つのDCT係数を保持する固定サイズアレイを選んだ。ここでも、線２２０－２は図１１Aの線２２０のCIFシーケンスに対応し、線２２２－２は図１１Aの線２２２のCIFシーケンスに対応している。この選択が、オーバーフローベクトルのサイズに最小にしてDCT係数を約２００に抑え、負荷因子を約９％と約１５％の間に維持する。当業者ならば、固定サイズアレイはブロックあたり８係数に限定されるものではなく、ブロックあたりの係数の数は適した任意の数を選んでいいことが明らかである。さらに、固定サイズアレイの個々のブロックは適した任意の構成にすることができる。例えば、８つの係数を保持する能力を有するブロックを、例えば、８ｘ１のブロック、４ｘ２のブロックとして配列することができる一方、９つの係数を保持する能力を有するブロックを、例えば、９ｘ１のブロック、３ｘ３のブロックとして配列することができる。

【００４７】図１２は、本発明の１実施例による、ビットストリームを復号化するのに要するメモリ所要量を減らすための方法のオペレーションのフローチャートである。この方法は、ビデオビットストリームを受け取るオペレーション２３０から始まる。１実施例で、ビットストリームは低レートビットストリームである。例えば、ビデオストリームを、H.263、Motion Pictures Expert

Group (MPEG-1/2/4)、H.261、Joint Photographic Expert Group (JPEG)など、ビデオ符号化規格と関連付けることができる。この方法は次に、オペレーション２３２に進み、そこでビットストリームのフレームが、そのフレームと関連付けられるデータの各ブロックの離散コサイン変換(DCT)領域表現に復号化される。ここで、ビデオは、図2、6B、１５に示したデコーダなど、デコーダの初めの２ステージで処理される。すなわち、ビデオデータは、圧縮されたビットストリームをDCT領域表現に復号化するために、可変長デコーダステージ及び反量子化ステージで処理される。なお、DCT領域表現は圧縮状態フォーマットになっている。フレームは一度に１ブロックずつ復号化される。この方法は次に、オペレーション２３４に進み、そこでDCT領域表現の非零係数が識別される。ここで、データブロックのDCT領域表現と関連付けられる６４のDCT係数のうち、６４中の比較的少数のDCT係数が概して非零係数である。

【００４８】引き続き、図１２において、この方法は次に、ハイブリッドデータ構造をアセンブルするオペレーション２３６に進む。ハイブリッドデータ構造は、固定サイズアレイと可変サイズオーバーフローベクトルとを含んでいる。模式的な一つのハイブリッドデータ構造は、図１０に示した複数の固定サイズブロック及び可変サイズオーバーフローベクトルを含む固定サイズアレイである。この方法は次に、オペレーション２３８に進み、そこでDCT領域表現の非零係数がハイブリッドデータ構造の中に挿入される。図１０において説明したように、ビデオデータブロックのDCT領域表現の非零係数は、固定サイズアレイ内の固定サイズブロックと関連付けられる。非零係数の数が、ビデオデータブロックと関連付けられる固定サイズブロックの容量を超えると、残りの非零係数は可変サイズオーバーフローベクトルに保持される。１実施例で、インデックス表は、オーバーフローベクトル内のデータを固定サイズアレイ内のしかるべき固定サイズブロックに写像する。従って、ハイブリッドデータ構造と非零係数の保持により、メモリ所要量が低減される。より具体的に言えば、ビデオ画質の損失なしにメモリ所要量を５０％低減することができる。

【００４９】なお、データフレームと関連付けられる各DCT領域表現の非零係数はハイブリッドデータ構造に保持される。保持されたフレームのデータは次に、表示するために組み合わされ、解凍される。１実施例では、その次のフレームがハイブリッドデータ構造に保持されるべくDCT領域表現に復号化されてしまえば、ハイブリッドデータ構造内で前のフレームと関連付けられたデータがフラッシュされる。以下にさらに説明するように、逆動き補償は圧縮領域の保持されたデータに対して行なわれる。逆動き補償は、完全画素逆動き補償には整数近似を用い、半画素逆動き補償には因数分解を用いる。

【００５０】空間H.263ビデオデコーダの主な構成要素には、ランレングス復号化、逆DCT、及び逆動き補償がある。タイミングプロファイラを用いて、1.1 GHzペンティアム(Pentium;登録商標)４プロセッサで走行しているTELENORのH.263ビデオデコーダの性能をベースラインデータで測定する。ベースラインデータを復号化し、システム呼び出しを無視して、プロファイラは、１４４のフレームを復号化するのに要する総体的な時間を測定すると共に、各構成要素のタイミング特性を詳細に記述する。表2は、空間的H. 263ビデオデコーダのタイミングプロファイルで、特に、選ばれた機能のタイミング結果を示すものである。

【表２】

機能	機能時間 (ms)	ヒット数
画像表示	772	144
逆動き補償	243	56336
ランレングス復号化	57	39830
逆DCT	3	42253

【００５１】表3は、最適化されていない圧縮領域H.263ビデオデコーダのタイミングプロファイルである。一つの模式的デコーダのパイプライン構成は図2に示したデコーダである。

【表３】

機能	機能時間 (ms)	ヒット数
逆動き補償	9194	56336
画像表示	1547	144
ランレングス復号化	32	39830
逆DCT	652	340197

【００５２】表2に示すように、空間領域ビデオデコーダは、１４４のフレームを復号化するのに約１．２秒かかる。この時間のほとんどが、例えば、WINDOW™など、適したオペレーティングシステムで表示するために、各

フレームのカラー値をYUVからRGBに変換する画像表示機能にとられている。ランレングス復号化、逆DCT、逆動き補償といった機能が、ビデオを復号化するのに要する時間全体のうちの約２５％をとっている。逆動き補償は空間領域で特に高速である。ここで、完全画素の動き補償は単にポインタをメモリもしくはフレームバッファ内の位置に設定してデータブロックをコピーするだけなのに対し、半画素の動き補償はメモリにポインタを設定し且つシフトオペレータを用いて値を補間する。対照して、表３は、最適化されていない圧縮領域ビデオデコーダのタイミング結果のいくつかを示したものである。最適化されていない圧縮領域デコーダは、同じ１４４のフレームを復号化するのに約１３．６７秒かかっている。

【００５３】

圧縮領域デコーダにとって主なボトルネックは逆動き補償機能である。先に方程式(８)で説明したように、圧縮領域での完全画素の逆動き補償には４つの(*TMi*)項がなければならない。ここで、*TMi* は、８ｘ８の行列ブロック F'_i に変換行列 C_{ij} を前後乗じることと定義される。

【数６】

$$F_k = TM_1 + TM_2 + TM_3 + TM_4 \qquad (10)$$

ここで、$TM_i = C_{i1}F'_iC_{i2}$ (11)

【００５４】

表４は、完全画素変換行列を定義している。ここで、C_{ij} は８ｘ８の DCT 行列を表わし、U_k 及び L_k は、先に方程式３～６に定義されている。

【表４】

完全画素変換行列	行列定義
$C_{11}=C_{21}$	SU_hS'
$C_{31}=C_{41}$	$SL_{8-h}S'$
$C_{12}=C_{32}$	SL_wS'
$C_{22}=C_{42}$	$SU_{8-w}S'$

【００５５】８ｘ８行列の乗算には各々、５１２の掛け算と４４８の加算が必要である。知られているように、行列乗算は計算的に高価である。表５は、マクロブロックの行列乗算、行列因数分解、共用ブロックなど最適化スキームと、図２、６B、１５に示したパイプラインなど圧縮領域ビデオパイプラインのハイブリッドスキームとの比較を示している。各データフレームが３５２本のラインを含み、ラインあたり２８８個の画素を有する共通中間フォーマットといったビデオフォーマットをサポートしているハンドヘルド機器で容認できる画質を提供するために、圧縮領域ビデオ復号化パイプラインは毎秒約１５～２５フレーム（fps）というレートで復号化すべきである。

【表５】

最適化	復号化時間(s)	フレーム数	FPS	備考
空間領域	9.79	144	14.71	オリジナルTELENOR H.263 ビデオデコーダ
行列ｘ行列	14.17	144	10.16	TMの完全８ｘ８行列乗算
近似	9.82	144	14.66	時間は良好だがPSNRは劣る
因数分解	12.95	144	11.12	PSNRは良好だが時間は劣る
共用ブロック	14.85	144	9.70	改善は見られない
ハイブリッド	9.83	144	14.65	時間、PSNR、共に良好

【００５６】圧縮領域ビデオ復号化パイプラインのエンハンスメントの一つは、ブロックアライメントにより、方程式（１０）のTMi演算数を減らすことである。ブロックアライメントを、例えば、次のように行なう：１列の１４４フレームを復号化して、ブロックアライメント率を全ブロックの３６．７％で測定する。図１３は、行列乗算を減らすブロックアライメントの３つの例を説明している概略図である。ブロックアライメント２４０（(w=8, h=4)）、ブロックアライメント２４２（(w=4, h=8)）、及びブロックアライメント２４４（(w=8, h=8)）の各ケースが描かれている。これらの例２４０、２４２、２４４では各々、対応するブロックとのオーバーラップがゼロになると、TMi演算はなくなる。但し、DCT領域（圧縮領域）で、ブロックアライメントは、半画素補間を指定している場合には、節約にはならない。圧縮領域における半画素動き補償の方程式は以下の通り。(w=8, h=8)の例で、半画素補間には、方程式１２及び１３に示されているように、４つのTMi演算が依然として必要である。表６は、半画素変換行列 C_{hpij} を定義するにあたり、参考までに提供しているものである。

【数７】

$$F_{hpk} = TM_{hp1} + TM_{hp2} + TM_{hp3} + TM_{hp4} \qquad (12)$$

$$TM_{hpi} = C_{hpi1} F'_i C_{hpi2} \qquad (13)$$

【表6】

半画素変換行列	水平方向補間	垂直方向補間	水平垂直方向補間
$C_{hp11} = C_{hp21}$	$SU_h S'$	$S(U_h + U_{h+1})S'$	$S(U_h + U_{h+1})S'$
$C_{hp31} = C_{hp41}$	$SL_{8-h} S'$	$S(L_{8-h} + L_{9-h})S'$	$S(L_{8-h} + L_{9-h})S'$
$C_{hp12} = C_{hp32}$	$S(L_w + L_{w+1})S'$	$SL_w S'$	$S(L_w + L_{w+1})S'$
$C_{hp22} = C_{hp42}$	$S(U_{8-w} + U_{9-w})S'$	$SU_{8-w} S'$	$S(U_{8-w} + U_{9-w})S'$

【００５７】完璧にアライメントされたブロックでも、半画素補間により近傍の画素とのオーバーラップが生じる。図１４は、完璧にアライメントされたDCTブロックの半画素補間の概略図である。半画素補間により、１画素幅及び１画素高だけ近傍ブロックへのオーバーラップが生じる。

【００５８】図２のデコーダの機能ブロックの並べ替えにより圧縮領域復号化パイプラインの処理スピードを上げることができる。表２及び３で、逆DCTブロックの処理時間は、圧縮領域（632 ms）と比べ、空間領域のほうがはるかに短い（3 ms）。空間領域で、逆DCTは、フィードバックループの前にイントラブロック及び誤差係数に対して用いられる。具体的に言えば、イントラブロック及び誤差係数は、ビデオの全ブロックの１５％未満にあたる。その他の８５％では、単に逆DCT機能は省かれる。圧縮領域では、逆DCTが、パイプラインの最終ステージで、ビデオの各フレームのブロック１００％に用いられる。

【００５９】図１５は、本発明の１実施例による、ビデオデータの処理を向上させる圧縮領域ビデオデコーダの機能ブロックの並べ替えを説明している概略図である。ここで、機能ブロックは並べ替えられ、圧縮領域パイプラインは２ヶ所で分けられる。最初のスプリットはVLD１２４及びDQ１２６の後に点（ｉ）２５２で起きる。上流のブランチで、パイプラインはメモリ圧縮１２８の内部DCT領域表現を保っている。下流のブランチで、パイプラインは、誤差係数を空間領域に復号化するために、RLD及びIDCTを前方に移動する。２番目のスプリットは、動き補償（MC）時に点（ii）２５４で発生する。動き補償時に、方程式（７）によると、空間領域出力が生成される可能性がある。ディスプレイ１３６に表示するために点（iii）２５６で現ブロックを再構築するべく、出力を誤差係数に直に加算することができる。内部DCT表現を維持するためにDCTブロック２５０がフィードバックループに挿入される。点（ｉ）２５２でのRLD１３２とIDCT１３４との組合せ及び点（ii）２５４でのDCTは、図２のパイプラインの最終ステージでのIDCTブロックと比べ、必要な計算が少ない。表７は、本書で説明している他の最適化スキームに加えて組合せ可能な図１５に示した並べ替えにより２０％のスピードアップを図れることを実証している。

【表７】

機能	ブロックの%	備考
図 15 の点(i)での IDCT	15%	イントラブロック及び誤差係数は全ブロックの小さな部分を表わす
図 15 の点(ii)での DCT	63%	アライメントされていないブロックは DCT が要るけれども、アライメントされたブロックは DCT せずに直接コピーされる
図 2 の IDCT	100%	DCT 領域で全ブロックに適用

【００６０】１実施例で、方程式（１１、１３）の基本的TM演算に必要な掛け算の数を減らすことにより逆動き補償が加速される。完全８ｘ８行列乗算を計算する代わりに、方程式１４に示すように、DCT行列Sが疎行列の列に因数分解される。方程式（１７）の疎行列は、順序行列($A_1, A_2, A_3, A_4, A_5, A_6$)及び対角行列(D,M)を含む。この

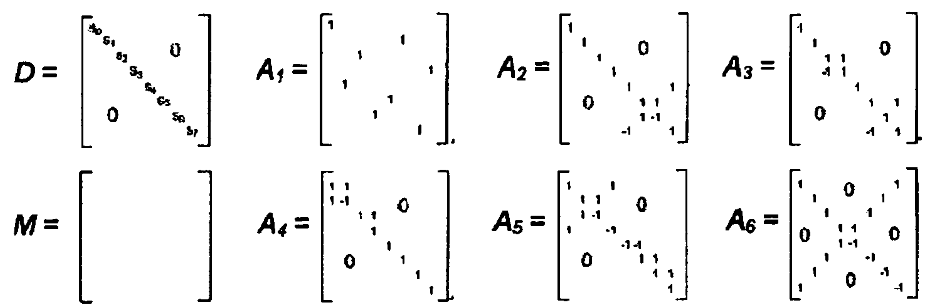
因数分解を方程式（１５）の中に代入すると、方程式（１６）のTM_iの完全に因数分解された方程式を導き出すことができる。これは、方程式（１１、１３）よりも掛け算の数が少ない。

【数８】

$$S = DA_1A_2A_3MA_4A_5A_6 \qquad (14)$$

$$TM_i = Sc_{i1}S^tF'_iSc_{i2}S^t \qquad (15)$$

$$TM_i = (DA_1A_2A_3MA_4A_5A_6)c_{i1}(DA_1A_2A_3MA_4A_5A_6)^tF'_i\,(DA_1A_2A_3MA_4A_5A_6)c_{i2}(DA_1A_2A_3MA_4A_5A_6)^t \qquad (16)$$



(17)

$$D = \mathrm{diag}\{0.3536, 0.2549, 0.2706, 0.3007, 0.3536, 0.4500, 0.6533, 1.2814\} \qquad (18)$$

$$A = 0.7071, B=0.9239, C=0.3827 \qquad (19)$$

【００６１】従って、行列乗算は、行列順列で置き換えられる。但し、方程式（１６）に示すように、項TM_i,の完全に因数分解された式は必ずしも逆動き補償をスピードアップするとは限らない。基本的に、掛け算がメモリアクセスとトレードされたので、メモリアクセスが多すぎると、実のところ復号化プロセスを遅くする。従って、これらの競合する機能間のバランスをとるべく行列の再グループ化が行なわれる。行列S（$= G_0G_1$）は２つの項、$G_0=DA_1A_2A_3$（順列と乗算の混合）と、$G_1=MA_4A_5A_6$（順列と加算の混合）とに因数分解される。方程式（２４）で逆動き補償に因数分解された式を形成するために、固定行列J_i，K_iが定義され、方程式（１０及び１２）の中に代入される。

【数９】

$$J_h = c_{11}G_1^t = c_{21}G_1^t, \quad J_w = G_1c_{12} = G_1c_{32} \qquad (20)$$

$$K_h = c_{31}G_1^t = c_{41}G_1^t, \quad K_w = G_1c_{22} = G_1c_{42} \qquad (21)$$

半画素補間でも同様に、

【数１０】

$$J_h = c_{hp11}G_1^t = c_{hp21}G_1^t, \quad J_w = G_1c_{hp12} = G_1c_{hp32} \qquad (22)$$

$$K_h = c_{hp31}G_1^t = c_{hp41}G_1^t, \quad K_w = G_1c_{hp22} = G_1c_{hp42} \qquad (23)$$

$$F_k = S[J_hG_0^tF'_1G_0J_w + J_hG_0^tF'_2G_0K_w + K_hG_0^tF'_3G_0J_w + K_hG_0^tF'_4G_0K_w]S^t \qquad (24)$$

【００６２】固定行列J_i，K_iによる高速乗算を実現することにより、スピードをさらに向上させることができる。固定行列には構造の繰り返しが入っている。例えば、行列J_6は次のように定義される。

【数11】

$$J_6 = \begin{bmatrix} 1 & -1 & -a & 0 & b & a & c & 0 \\ 1 & 1 & -a & -1 & b & 0 & c & 0 \\ 1 & 1 & -a & -1 & -b & 0 & -c & 0 \\ 1 & -1 & -a & 0 & -b & -a & -c & 0 \\ 1 & -1 & a & 0 & c & -a & -b & 0 \\ 1 & 1 & a & 1 & c & 0 & -b & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

【0063】ここでは、a=0.7071、b=0.9239、c=0.3827とする。u= $\{u_1, \cdots, u_8\}$及びv=$\{v_1, \cdots, v_8\}$と仮定すると、u = J_6vを計算するには、方程式の列を次のステップに従って計算する：

【数12】

$$y_1 = v_1 + v_2 \quad (25)$$

$$y_2 = v_1 - v_2 \quad (26)$$

$$y_3 = av_3 \quad (27)$$

$$y_4 = av_6 \quad (28)$$

$$y_5 = y_1 - y_3 \quad (29)$$

$$y_6 = y_5 - y_4 \quad (30)$$

$$y_7 = y_3 - y_4 \quad (31)$$

$$y_8 = y_3 + y_4 \quad (32)$$

$$y_9 = (b+c)(v_5 + v_7) \quad (33)$$

$$y_{10} = cv_5 \quad (34)$$

$$y_{11} = bv_7 \quad (35)$$

$$y_{12} = y_9 - y_{10} - y_{11} \quad (36)$$

$$y_{13} = y_{10} - y_{11} \quad (37)$$

$$u_1 = y_2 - y_7 + y_{12} \quad (38)$$

$$u_2 = y_6 + y_{12} \quad (39)$$

$$u_3 = y_6 - y_{12} \quad (40)$$

$$u_4 = y_2 - y_8 - y_{12} \quad (41)$$

$$u_5 = y_2 + y_7 + y_{13} \quad (42)$$

$$u_6 = y_1 + y_3 + u_4 + y_{13} - u_8 \quad (43)$$

$$u_7 = 0 \quad (44)$$

$$u_8 = 0 \quad (45)$$

【0064】

このように、行列ベクトル乗算を方程式の列に変換した。上に示した方程式の列は、５回の乗算と２１回の加算を必要とする。方程式（２４）の行列乗算$J_kG_o^tF^t$では、１０４回の乗算及び１６４回の加算が必要である。従って、方程式（８）の行列乗算$C_{11}F^t$に必要な掛け算の回数と比べ、5倍減らすことができる。その上、３２ビットの浮動小数点算術を用いるこの行列演算時に精度が失われることはない。しかしながら、表５において、因数分解は、圧縮領域パイプラインのスピードを行列間に比べ９％しか向上させていない。従って、余分のメモリアクセスがフレームレートを約１５fps から約２５fps という目標レート以下に低下させるので、因数分解だけでは事足りない。

【００６５】逆動き補償の更なるスピードアップを図るために、方程式（１１、１３）の基本的なＴＭ演算に必要な掛け算を削除する。完全画素及び半画素の行列、C_{ij}及びC_{hpij}が、2^{-5}のベキに一番近い２進数に近似される。これらの行列を２進数に近似すると、方程式（１０、１２）の逆動き補償を解くために、右シフトや加算など基本的な整数演算を用いることによって行列乗算を行なうことができる。例えば、h=1の場合、完全画素行列C_{11}を以下のように調べる。なお、その他の行列は同じように近似されている。

【数１３】

$$C_{11}=\begin{bmatrix} 0.12501651 & -0.17338332 & 0.16332089 & \cdots & -0.03447659 \\ 0.17340284 & -0.24048958 & 0.22653259 & \cdots & -0.04782041 \\ 0.16334190 & -0.22653624 & 0.21338904 & \cdots & -0.04504584 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0.03449197 & -0.04783635 & 0.04506013 & \cdots & -0.00951207 \end{bmatrix} \quad (46)$$

【数１４】

行列の各要素を２のベキに一番近い値に丸めると、行列（４７）が生まれる。

$$\hat{C}_{11}=\begin{bmatrix} 0.1250 & -0.1875 & 0.1875 & \cdots & -0.0625 \\ 0.1875 & -0.2500 & 0.2500 & \cdots & -0.0625 \\ 0.1875 & -0.2500 & 0.1875 & \cdots & -0.0625 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0.0625 & -0.0625 & 0.0625 & \cdots & 0 \end{bmatrix} \quad (47)$$

【００６６】DCT要素は[-2048 to 2047]の範囲内に入っているから、DCT係数を直接シフトすることでほとんどの値がゼロに駆動される。中間結果の精度を維持するために、復号化パイプライン全体で各DCT係数を2^8でスケーリングする。この倍率は、量子化及び反量子化のステップで導入されるから、余分な演算は発生しない。

【００６７】さらに、積和のルールに従って項をグループ化することにより高速行列乗算を実現する（方程式（４８～５０）を参照）。

【数１５】

$$u_1=0.1250v_1-0.1875v_2+0.1875v_3-0.1250v_4+0.1250v_5-0.1250v_6+0.0625v_7-0.0625v_8 \quad (48)$$

$$u_1=(v_1>>3)-(v_2>>3)-(v_2>>4)+(v_3>>3)+(v_3>>4)-(v_4>>3)+(v_5>>3)-(v_6>>3)+(v_7>>4)-(v_8>>4) \quad (49)$$

$$u_1=(v_1-v_2+v_3-v_4+v_5-v_6)>>3+(-v_2+v_3+v_7-v_8)>>4 \quad (50)$$

【００６８】

$u=\{u_1,\cdots,u_8\}$及び$v=\{v_1,\cdots,v_8\}$と仮定すると、$u=\hat{C}_{11}v$の計算は次のように算出することができる：

【数16】

$$u_1 = (v_1 - v_2 + v_3 - v_4 + v_5 - v_6) >> 3 + (-v_2 + v_3 + v_7 - v_8) >> 4 \quad (51)$$

$$u_2 = (v_3 - v_2) >> 2 + (v_1 - v_4 + v_5 - v_6 + v_7) >> 3 + (v_1 - v_4 + v_5 - v_8) >> 4 \quad (52)$$

$$u_3 = (v_1 + v_3 - v_4 + v_5 - v_6) >> 3 - (v_2 >> 2) + (v_1 + v_3 - v_4 + v_5 + v_7 - v_8) >> 4 \quad (53)$$

$$u_4 = (v_1 - v_2 + v_3 - v_4 + v_5 - v_6) >> 3 + (v_3 - v_2 - v_4 + v_7 - v_8) >> 4 \quad (54)$$

$$u_5 = (v_1 - v_2 + v_3 - v_4 + v_5 - v_6) >> 3 + (-v_2 + v_3 + v_7 - v_8) >> 4 \quad (55)$$

$$u_6 = (v_1 - v_2 + v_3 - v_4 + v_5) >> 3 + (v_7 - v_6) >> 4 \quad (56)$$

$$u_7 = (v_1 + v_3 - v_4 + v_5 - v_6 + v7) >> 4 + (v_2) >> 3 \quad (57)$$

$$u_8 = (v_1 - v_2 + v_3 - v_4 + v_5) >> 4 \quad (58)$$

【0069】

行列近似には、合わせて１７回の右シフトと５７回の加算が必要である。方程式（８）の行列近似$\hat{C}_{\mu}F'$では、１３６回の右シフトと４５６回の加算を行なわなければならない。従って、浮動小数点の精度を用いると、行列乗算と比べ計算量がかなり減る。実際、表５から、近似の技法により圧縮領域パイプラインが３１％スピードアップしていることが明らかである。これは、約１５fps の目標フレームレートを達成するのに十分である。但し、サンプルビデオのＰＳＮＲは低下し、中程度の動きのエリアでドリフトが目に見える。

【００７０】ビデオの動きに基づいて選択的に適用される変換行列TMのハイブリッド因数分解／整数近似は、容認できる画質を維持しながら、同時に、好ましい約１５～２５fpsのフレームレートを実現する。先に触れたように、整数近似技法は、デコーダの計算量を低減するだけでなく、復号化されたビデオのPSNRも低下させる。同時に、因数分解技法では、良好なPSNRは維持できるけれども、好ましいフレームレートを満たすにはデコーダの計算量は低下しない。整数近似技法の計算量に低さと因数分解技法の精度の高さとを統合することで、低レートビデオビットストリームをサポートするための圧縮領域ビデオ復号化パイプラインを得ることができる。

【００７１】２つのタイプの変換行列、つまり、方程式（１１）に示した完全画素動き補償TM_iと、方程式（１３）に示した半画素動き補償TM_{hpi}について説明してきた。TM_iに近似行列を用いると、完全画素動き補償は、８ｘ８浮動小数点行列を用いた場合と比べ、計算量が２８％ですむ。しかしながら、半画素変換行列TM_{hpi}に近似技法を直接用いる場合、TM_{hpi}の近似行列を用いると、半画素動き補償はＰＳＮＲを低下させる（表８）と共に復号化されたビデオに目に見える歪みを発生させることが観察された。誤差源は２つある。その一つは、半画素変換行列TM_{hpi}が近似技法により敏感に反応することである。表８で、TM_{hpi}はTM_i以外にも数多くの項からなる複合行列である。２つ目は、先に図６A及び６Ｂにおいて説明したように、半画素補間時の非線形処理が、近似技法によって発生した誤差と組み合わされて、中程度から高度の動き領域で目に見える誤差の累積を生じさせる。

【００７２】半画素行列に因数分解技法を選択的に適用することでこうした誤差の問題を解決することができる。先に説明したように、因数分解技法は浮動小数点の精度を維持するので、上に説明した誤差を最小限にすることができる。例えば、因数分解技法は、TM_{hpi}を有する行列乗算を、方程式（２５～４５）に示したのと同じような方程式の列に還元する。これらの方程式は３２ビット浮動小数点の精度を維持するので、近似誤差が生まれない。さらに、因数分解技法は動き補償時にDCTブロックを空間領域に復号化するので、図１５において説明した最適化をいま説明した最適化と組み合わせることが

できる。表5はハイブリッド法で15fpsの目標フレームレートを達成できることを実証しているのに対し、表8はハイブリッド法のPNSRは容認できるPSNRを実現することを示している。

【表8】

ビデオ (128 kbps, QCIF, 15 fps)	因数TMを有する圧縮領域 (PSNR_Y)	ハイブリッドTMを有する圧縮領域 (PSNR_Y)	近似TMを有する圧縮領域(PSNR_Y)
Sample A	25.53	25.53	22.65
Sample B	22.47	19.57	18.75
Sample C	30.79	30.66	29.90
Sample D	33.29	33.25	28.93
Sample E	31.27	31.10	28.89

【0073】図16は、本発明の1実施例による、圧縮領域で逆動き補償を行なうための方法のオペレーションのフローチャート図である。この方法は、圧縮されたビットストリーム内のビデオデータフレームを受け取るオペレーション260から始まる。1実施例で、ビットストリームは低レートビットストリームである。例えば、ビットストリームは、MPEG 4、H.263、H.261など、公知のビデオ符号化規格と関連付けられていて構わない。この方法は次にオペレーション262に進み、そこでビットストリームのフレームのブロックが離散コサイン変換（DCT）領域表現に復号化される。ここで、ビデオは、図2、6B、15に示したデコーダなど、デコーダの最初の2ステージで処理される。すなわち、ビデオデータは、圧縮ビットストリームをDCT領域表現に復号化するために、可変長デコーダのステージ及び反量子化のステージで処理される。なお、DCT領域表現は圧縮状態のフォーマットになっている。この方法は次にオペレーション264に進み、そこでDCT領域表現と関連付けられるデータがハイブリッドデータ構造に保持される。適したハイブリッドデータ構造は、図10及び12において説明したハイブリッドデータ構造である。1実施例で、ハイブリッドデータ構造は、例えば、セルラー電話、PDA、ウェブタブレット、ポケットパソコンなど、ビデオデータを表示するためのディスプレイ画面を有する携帯用電子機器のメモリ所要量を低減する。

【0074】引き続き、図16において、この方法はオペレーション266に進み、そこでDCT領域表現と関連付けられるデータに圧縮領域で逆動き補償が実行される。ここで、逆動き補償には、表5及び9において説明したハイブリッド因数分解／整数近似技法が選択的に適用される。この方法は次にオペレーション268に進み、そこでハイブリッド因数分解／整数近似技法が、処理中のビデオデータのブロックと関連付けられる変換行列のタイプを識別する。1実施例では、いま復号化されているビットストリームのビットセットの中の情報により変換行列のタイプが検出される。変換行列が半画素行列の場合には、この方法はオペレーション270に進み、そこでビットストリームを復号化するために因数分解技法が用いられる。1実施例では、先に方程式25～45において説明したように、因数分解技法により行列乗算が一連の方程式に還元される。すなわち、行列乗算が行列順列で置き換えられる。決定のオペレーション268で変換行列が完全画素行列であると判定された場合には、この方法はオペレーション272に進み、そこでビットストリームを復号化するために整数近似技法が用いられる。ここでは、方程式46～58において先に説明したように、逆動き補償を解くために、基本的な整数演算を用いて行列乗算を実行することができる。従って、容認できる画質を有しながら先に説明したハイブリッドデータ構造により達成できたメモリの低減を可能にする程度のフレームレートを実現するために、ハイブリッド因数分解／整数近似技法を選択的に適用することにより、圧縮領域での処理が実行される。

【0075】図17は、本発明の1実施例による、ハイブリッド因数分解／整数近似技法の選択的適用の概略図である。ディスプレイ画面280は低ビットレートビデオによって定義された画像を表示するように構成されている。例えば、ディスプレイ画面280を、PDA、セルラー電話、ポケットパソコン、ウェブタブレットなど、携帯用電子機器と関連付けることができる。ボール282はビデオで垂直方向に移動している。ブロック284は移動するオブジェクトの周囲に位置し、高度又は中程度の動き領域と考えられ、フレームからフレームで変わる。ブロック286はバックグラウンドを表わし、フレームからフレームで実質的に同じままである。従って、圧縮ビットストリームの復号化時に、データフレームのブロック284は、フレームからフレームで、高度な動き領域と関連付けられるのに対し、ブロック286はフレームからフレームで実質的に同じままである。高度動き領域と関連付けられるブロック284は、複合化の技法、つまり、因数分解時には、より高度な精度を必要とするが、ブロック286は実質的に変わらないので計算量の低い補間法、つまり、整数近似で許される。従って、因数分解技法を高度から中程度の動き領域ブロック

２８４に適用し、整数近似法をバックグラウンドブロック２８６に適用する。先に説明したように、ブロックが高度の動きと関連付けられるかどうか、つまり、因数分解による半画素動き補償を適用するかどうか、もしくはブロックがバックグラウンドデータかどうか、つまり、整数近似による完全画素動き補償を適用するかどうかを判定するために、ビットストリームに埋め込まれた情報が検出される。１実施例では、図２、６Ｂ、１５に示した動きベクトルが、動き補償が半画素か又は完全画素の動き補償かどうかを指定している。

【００７６】なお、先に説明した実施例はソフトウェアで実行しても、ハードウェアで実行しても構わない。当業者ならば、デコーダを、先に説明した機能を実現できるように構成された論理ゲートを含む半導体チップとして実施できることが分かる。例えば、ビデオデコーダをハードウェアで実現するには、本書で説明した必要な機能を実現するための論理ゲートのレイアウト及びファームウェアを合成するために、例えば、VERILOGなど、ハードウェア記述言語（HDL）を採用することができる。

【００７７】図１８は、本発明の１実施例による、メモリ所要量を最小限にするためのハイブリッドデータ構造を活用すると共に、ビットストリームデータを効率よく復号化するためのハイブリッド因数分解／整数近似技法を適用するように構成されたデコーダ回路構成を有する携帯用電子機器の簡約概略図である。携帯用電子機器２９０は、中央処理機構（CPU）２９４、メモリ２９２、ディスプレイ画面１３６、デコーダ回路構成２９８を含み、これらは全てバス２９６で互いに通信し合っている。デコーダ回路構成２９８は、先に説明したビデオ処理並びに圧縮領域で逆動き補償を実行するのに要するメモリ所要量を低減する機能を提供できるように構成された論理ゲートを含む。当業者ならば、デコーダ回路構成２９８はデコーダ回路構成が入っているチップ上にメモリを有していても、或いはメモリはチップの外に配置されていても構わないことが明らかである。

【００７８】図１９は、本発明の１実施例による、図１８のデコーダ回路構成のより詳細な概略図である。入ってくるビットストリーム１２２は、デコーダ２９８の可変長デコーダ（VLD）回路構成３００によって受け取られる。当業者ならば、デコーダ回路構成２９８をプリント配線板上に配置された半導体チップ上に設置して構わないことが分かる。ＶＬＤ回路構成３００は、動き補償回路構成３０６に動きベクトル信号を供給する。ビデオ処理メモリ３０８は、圧縮領域で反量子化回路構成３０２からのビデオの内部表現を保持する。DCT回路構成３０４は動き補償回路構成３０６からのビデオの内部ＤＣＴ表現を維持する。ランレングス復号（RLD）回路３１０及び逆離散コサイン変換（IDCT）回路構成３１２は、ディスプレイ画面１３６に表示できるようにビデオデータを解凍する。なお、ここで説明している回路構成のブロックは、図２、６Ｂ、１５において説明したブロック／ステージと同じような機能を提供する。

【００７９】

【発明の効果】要約すると、今まで説明してきた発明は、ビデオメモリ量を減らし、圧縮領域で逆動き補償を実行する圧縮領域ビデオデコーダを提供するものである。現フレームを定義するために基準フレームの非零DCT係数を保持及び操作するように構成されたハイブリッドデータ構造によりメモリ低減を実現する。ハイブリッドデータ構成は、ビデオデータフレームの各ブロックと関連付けられる固定サイズブロックを有する固定サイズアレイを含んでいる。固定サイズブロックの容量を超える非零係数を収容できるように、ハイブリッドデータ構造には可変サイズオーバーフローベクトルが備わっている。圧縮領域ビデオデコーダにより達成されたメモリ圧縮量は、空間領域ビデオデコーダと比べ、上限で2倍である。圧縮領域ビデオデコーダの逆動き補償は、容認できる画質のビデオで毎秒約１５～２５フレームを実現できるように最適化される。復号中のブロックにハイブリッド因数分解／整数近似が選択的にかけられる。因数分解／整数近似技法のうちのどの補間を適用するかを判定する基準は、変換行列に基づく。つまり、半画素行列には因数分解が適用されるのに対し、完全画素行列には整数近似が適用される。なお、１実施例では、本書で説明した圧縮領域パイプラインをＭＰＥＧ－４のシンプルプロファイルビデオデコーダに取り入れることができる。さらに、実施例により、例えば、電池で動く（CPU制約型）機器のパワースケーラブル復号化やビデオ会議システムの複合化など、多種多様なアプリケーションを追求できるようになる。

【００８０】上述の実施例を考慮すれば、本発明は、コンピュータシステムに保持されたデータを必要とするいろいろなコンピュータで実行されるオペレーションを採用することができることが分かる。こうしたオペレーションには、物理的数量の物理的操作が含まれる。必ずしもそうとは限らないが、普通、こうした数量は、保持、変換、結合、比較、さもなければ操作の対象となり得る電気信号又は磁気信号の形をとる。さらに、実行される操作は、生成、識別、判定、又は比較といったような言葉で示されることが多い。

【００８１】以上説明した本発明は、ハンドヘルド機器、マイクロプロセッサシステム、マイクロプロセッサベースの或いはプログラマブル消費者向けエレクトロニクス、ミニコンピュータ、メインフレームコンピュータなど、その他のコンピュータシステム構成と共に実施することができる。本発明は、通信ネットワークを経由してリンクされている遠隔処理装置によってタスクが実行される分散型コンピューティング環境において実施することができる。

【００８２】本発明は、コンピュータ可読媒体上のコン

ピュータ可読コードとして実施することもできる。コンピュータ可読媒体は、データを保持し、後からそのデータをコンピュータシステムが判読できればどんなデータ記憶装置でも構わない。コンピュータ可読媒体の例としては、ハードドライブ、ネットワーク接続記憶装置（NAS）、読み取り専用メモリ、ランダムアクセスメモリ、CD-ROM、CD-R、CD-RW、磁気テープや、その他の光学式および非光学式データ記憶装置がある。分散してコンピュータ可読コードを保持及び実行されるように、コンピュータ可読媒体をネットワークでつながったコンピュータシステムで分散することもできる。コンピュータ可読媒体は、コンピュータコードを含んだ電磁搬送波でも構わない。

【００８３】以上、本発明を、はっきり理解できるように、より詳細に説明してきたが、添付した特許請求の範囲内で変更、修正が可能なことは明らかである。従って、本願の実施例は説明のためで、制約するためのものではない。また、本発明は、本書で説明した詳細に限定されるものではないけれども、特許請求の範囲内及び特許請求と同等の範囲内で修正可能である。請求項において、要素及び／又は工程は、請求項にはっきりと明記されていない限り、何らオペレーションの特定の順序を暗に示しているものではない。

【図面の簡単な説明】

本発明は、添付の図面と共に以下に述べる詳細な説明により容易に理解できるだろう。類似した構造上の要素を類似の参照番号で示している。

【図１】　ビデオデータを復号化すると共に空間領域で動き補償を実行するためのビデオデコーダの概略図。

【図２】　本発明の１実施例による、逆動き補償が圧縮領域で実行されるように構成されたビデオデコーダの概略図。

【図３】　空間領域で実行される逆動き補償を説明している概略図。

【図４】　H.263規格と関連付けられる強制更新メカニズムの有効性を実証するために、複数のフレームのピークの信号対雑音比（PSNR）を説明しているグラフ。

【図５】　H.263規格における半画素値の判定を説明している概略図。

【図６】　Ａ：ベースライン空間ビデオデコーダの概略図。Ｂ：本発明の１実施例による、圧縮領域ビデオデコーダの概略図。

【図７】　本発明の１実施例による、ビデオ符号化及び復号化プロセス時のブロック変換を説明しているブロック図。

【図８】　ランレングス表現における各８ｘ８ブロックの開始位置を見出すために個別のインデックスの使用を説明している概略図。

【図９】　Ａ、Ｂ：それぞれ、アレイベースのデータ構造及びリストデータ構造で、予測に予測誤差を加算するために必要な類別及び併合操作を説明する図。

【図１０】　本発明の１実施例による、メモリ圧縮及び計算効率性を可能にするアレイ構造及びベクトル構造を含むハイブリッドデータ構造の概略図。

【図１１】　Ａ、Ｂ、Ｃ：本発明の１実施例による、ハイブリッドデータ構造の固定サイズアレイの固定サイズブロックの容量及びオーバーフローベクトルを判定する際に評価される因子を説明しているグラフ。

【図１２】　本発明の１実施例による、ビットストリームを復号化するためのメモリ所要量を低減する方法のオペレーションを説明しているフローチャート。

【図１３】　行列乗算を減らすためのブロックアライメントの３つの例を説明している概略図。

【図１４】　完璧にアライメントされたDCTブロックの半画素補間の概略図。

【図１５】　本発明の１実施例による、ビデオデータの処理を向上させる圧縮領域ビデオデコーダの機能ブロックの並べ替えを説明する概略図。

【図１６】　本発明の１実施例による、圧縮領域で逆動き補償を実行するための方法のオペレーションを示すフローチャート。

【図１７】　本発明の１実施例による、ハイブリッド因数分解／整数近似技法の選択的な適用の概略図。

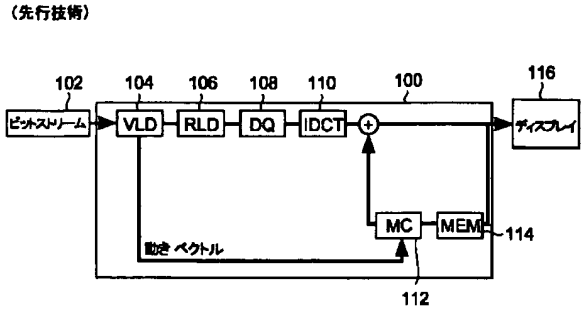
【図１８】　本発明の１実施例による、メモリ所要量を最小限にするためにハイブリッドデータ構造を活用すると共にビットデータストリームを効率的に復号化するためにハイブリッド因数分解／整数近似技法を適用するように構成されたデコーダ回路構成を有する携帯用電子機器の簡約概略図。

【図１９】　本発明の１実施例による、図１８のデコーダ回路構成のより詳細な概略図。

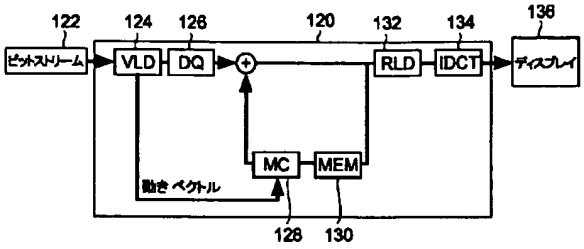
【符号の説明】

１００，１２０　デコーダ
１０２，１２２　ビットストリーム
１０４，１２４　可変デコーダ
１０６，１３２　ランレングスデコーダ
１０８，１２６　反量子化
１１０，１３４　逆離散コサイン変換
１１２，１２８　動き補償
１１４，１３０　メモリ
１１６，１３６　ディスプレイ
１６０　半画素補間

【図1】

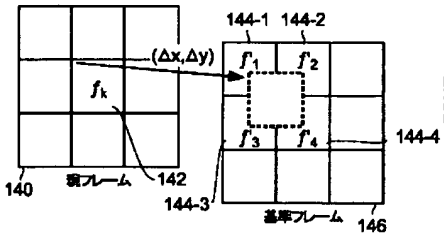


【図2】

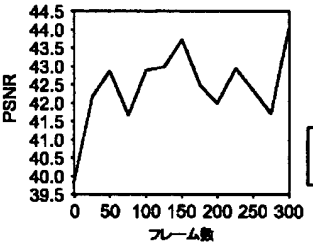


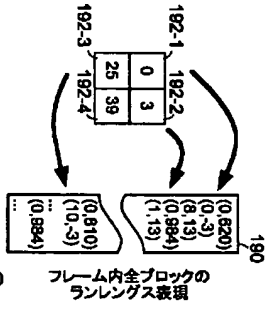
【図8】

【図3】

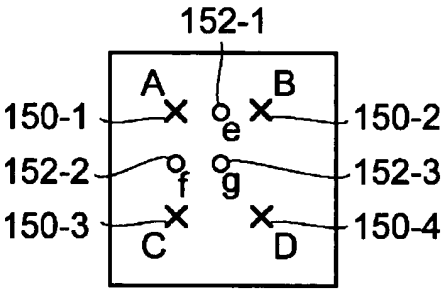


【図4】

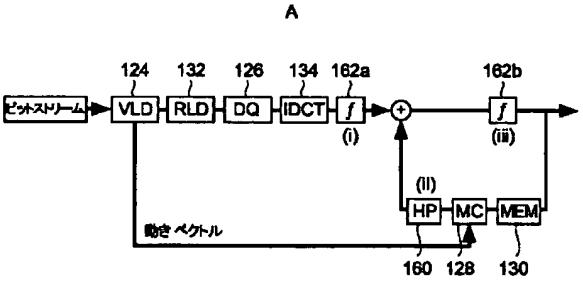




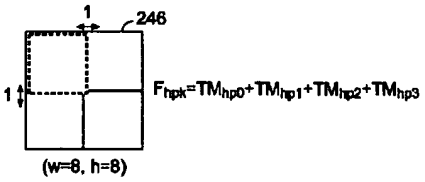
【図5】

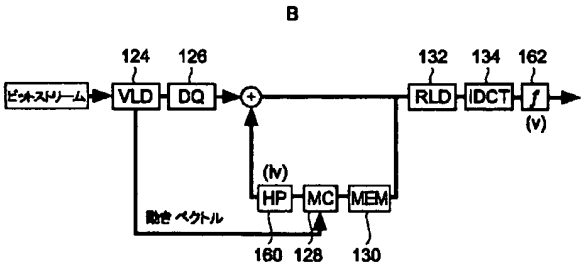


【図6】

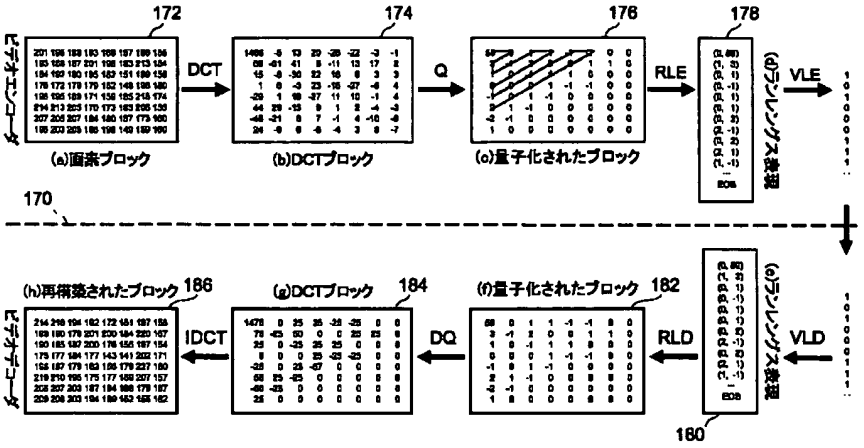


【図14】



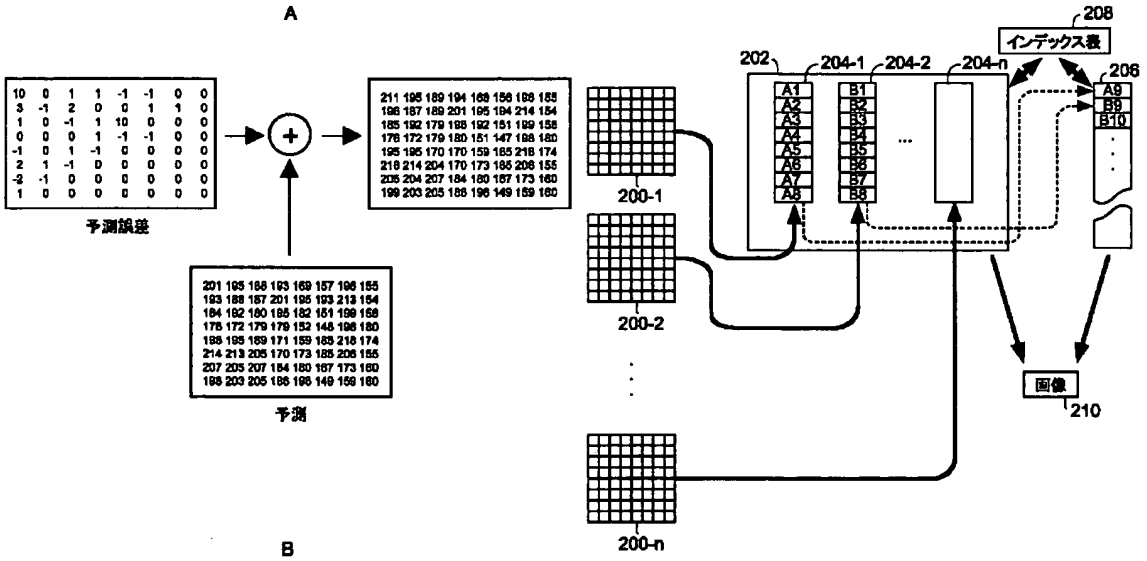


【図7】

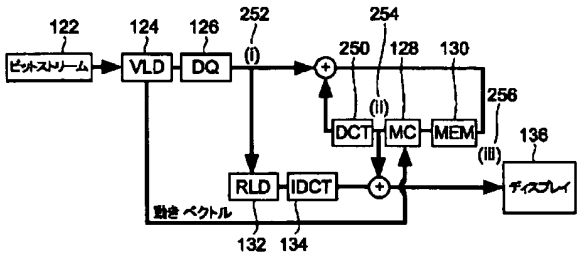


【図9】

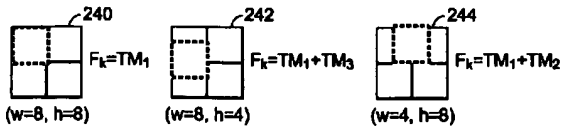
【図10】



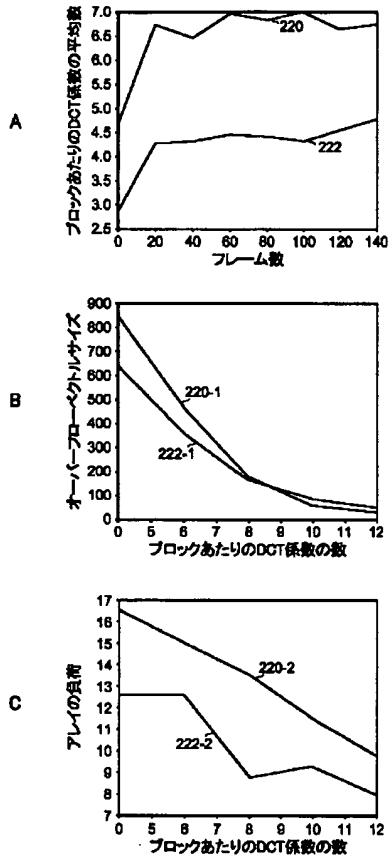
【図15】



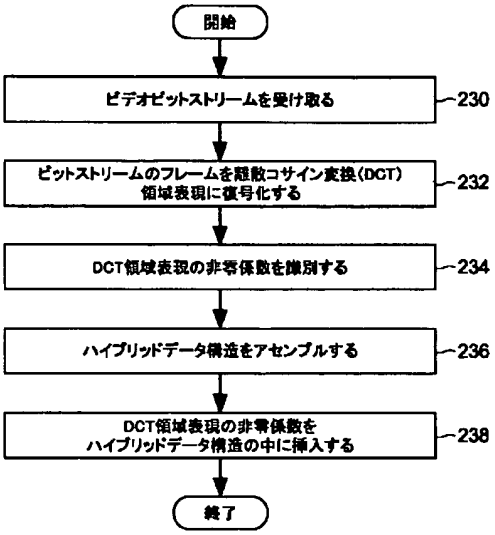
【図13】



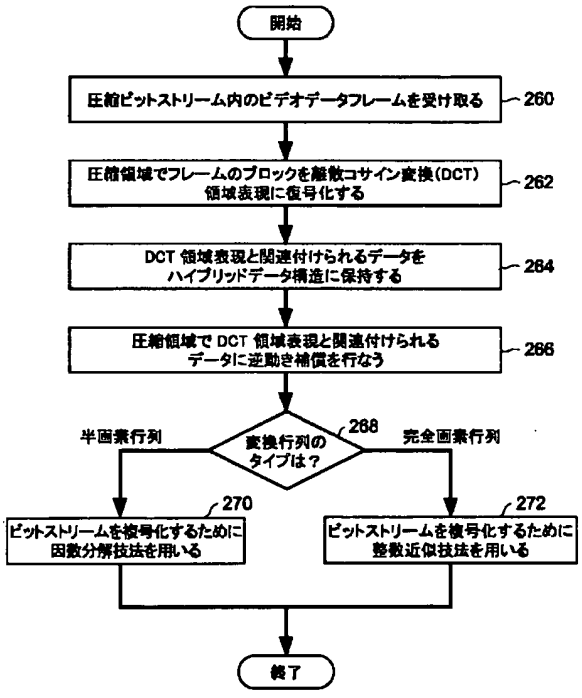
【図１１】



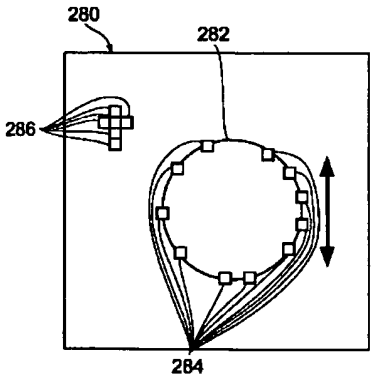
【図１２】



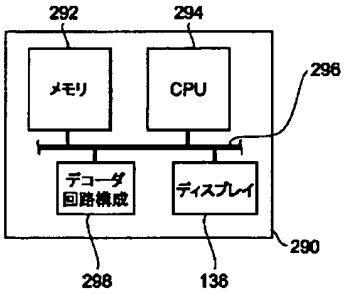
【図１６】



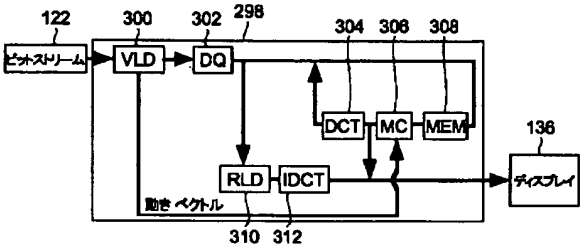
【図１７】



【図18】



【図19】



フロントページの続き

(72)発明者 ヴァスデヴ バスカラン
アメリカ合衆国 カリフォルニア州 サニーベール ノース マーフイ アベニュ 190

Fターム(参考) 5C059 KK08 MA00 MA05 MA23 MC11 MC38 ME05 NN15 NN21 PP04 SS10 SS20 TA61 TC00 UA05 UA33
5J064 AA03 BA09 BB05 BC01 BC02 BC08 BC09 BC14 BC23 BD03

【外国語明細書】

1. Title of Invention

Method and Apparatus for Memory Efficient Compressed Domain Video Processing and for Fast Inverse Motion Compensation Using Factorization and Integer Approximation

2. Claims

1. A method for reducing the memory requirements for decoding a bit stream, comprising:

receiving a video bit stream;

decoding a frame of the bit stream into a transform domain representation;

identifying non-zero coefficients of the transform domain representation;

assembling a hybrid data structure including a fixed size array and a variable size overflow vector; and

inserting the non-zero coefficients of the transform domain representation into the hybrid data structure.

2. The method of claim 1, wherein the video bit stream is a low rate video bit stream.

3. The method of claim 1, wherein the method operation of decoding a frame of the bit stream into a transform domain representation includes,

processing the bit stream through a variable length decoder and a dequantization block.

4. The method of claim 1, wherein the fixed size array includes fixed size blocks.

5. The method of claim 4 wherein the fixed size blocks are configured to store 8 non-zero coefficients of the transform domain representation.

6. The method of claim 1, wherein the method operation of inserting the non-zero coefficients of the transform domain representation into the hybrid data structure includes,

mapping coefficients in the fixed size array to corresponding coefficients in the variable size overflow vector for each block of the frame.

7. A method for decoding video data, comprising:

receiving a frame of video data within a compressed bit stream;

decoding a block of the frame into a transform domain representation in the compressed domain;

defining a hybrid data structure;

storing data associated with the transform domain representation in the hybrid data structure;

performing inverse motion compensation on the data associated with the transform domain representation in the compressed domain; and

after performing the inverse motion compensation on the data, decompressing the data for display.

8. The method of claim 7, wherein the hybrid data structure includes a fixed size array of fixed size blocks and a variable size overflow vector.

9. The method of claim 7, wherein the method operation of storing data associated with the transform domain representation in the hybrid data structure includes,

identifying non-zero coefficients of the transform domain representation;

storing the non-zero coefficients into a fixed size block of the fixed size array of the hybrid data structure until a capacity of the fixed size blocks is reached; and

after reaching the capacity of the fixed size blocks, storing non-zero c
oefficients exceeding the capacity of the fixed size blocks in an overfl
ow vector.

10. The method of claim 7, wherein the compressed bit stream is a low ra
te bit stream.

11. The method of claim 7, wherein the method operation of performing in
verse motion compensation on the data associated with the transform doma
in representation in the compressed domain includes,

applying a hybrid factorization and integer approximation technique to t
he data associated with the transform domain representation.

12. A computer readable media having program instructions for rearrangin
g low rate bit stream data for storage into a hybrid data structure, com
prising:

program instructions for identifying non-zero transform coefficients ass
ociated with a coded block of a frame of data;

program instructions for arranging the non-zero transform coefficients i
nto a fixed size array;

program instructions for determining if a quantity of the non-zero trans
form coefficients exceed a capacity of the fixed size array;

program instructions for storing the non-zero transform coefficients exc
eeding the capacity of the fixed size array in a variable size overflow
vector; and

program instructions for translating the non-zero transform coefficients
from a compressed domain to a spatial domain.

13. The computer readable media of claim 12, wherein the fixed size arra
y includes a plurality of fixed size blocks.

14. The computer readable media of claim 13, wherein each of the fixed s
ize blocks are configured to store a maximum of eight non-zero transform
coefficients.

15. The computer readable media of claim 12, further including:
program instructions for mapping coefficients in the fixed size array to corresponding coefficients in the variable size overflow vector for each block of the frame of data.
16. The computer readable media of claim 12, further including:
program instructions for performing inverse motion compensation on the non-zero transform coefficients through the application of a hybrid factorization and integer approximation technique.
17. A circuit, comprising:
a video decoder integrated circuit chip, the video decoder integrated circuit chip including,
circuitry for receiving a bit stream of data associated with a frame of video data;
circuitry for decoding the bit stream of data into a transform domain representation;
circuitry for arranging non-zero transform coefficients of the transform domain representation in a hybrid data structure in a memory associated with the video decoder; and
circuitry for decompressing the non-zero transform coefficients of the transform domain representation for display.
18. The circuit of claim 17, wherein the bit stream is a H.263 bit stream.
19. The circuit of claim 17, wherein the memory is separate from the video decoder integrated circuit chip.
20. The circuit of claim 17, further including:
circuitry for performing inverse motion compensation through a hybrid factorization and integer approximation technique.
21. The circuit of claim 17, wherein the memory is a static random access memory.

22. A device configured to display a video image, comprising:

a central processing unit (CPU);

a random access memory (RAM);

a display screen configured to present an image;

decoder circuitry configured to transform a video bit stream into a transform domain representation, the decoder circuitry capable of arranging non-zero transform coefficients of the transform domain representation in a hybrid data structure in a memory associated with the decoder circuitry, the decoder circuitry including circuitry for selectively applying a hybrid factorization/integer approximation technique during inverse motion compensation; and

a bus in communication with the CPU, the RAM, the display screen and the decoder circuitry.

23. The device of claim 22, wherein the device is a potable electronic device.

24. The device of claim 23, wherein the portable electronic device is selected from the group consisting of a personal digital assistant, a cellular phone, a web tablet and a pocket personal computer.

25. The device of claim 22, wherein the hybrid data structure includes a fixed size array having a plurality of fixed size blocks and a variable size overflow vector.

26. The device of claim 25, wherein each of the plurality of fixed size blocks are configured to hold 8 non-zero transform coefficients.

27. The device of claim 26, wherein non-zero transformcoefficients in excess of 8 are stored in the variable size overflow vector.

28. The device of claim 22, wherein the decoder circuitry includes an on-chip memory configured to store data associated with the hybrid data structure.

29. The device of claim 22, wherein the circuitry for selectively applyi

ng a hybrid factorization/integer approximation technique during inverse motion compensation includes,

circuitry for identifying blocks of a frame of the video image as being associated with one of an active motion area and an inactive motion area; and circuitry for performing inverse motion compensation by applying a factorization technique to the blocks associated with the active motion area and an integer approximation technique to the blocks associated with the inactive motion area.

30. The device of claim 22, wherein the video bit stream is a low rate video bit stream.

31. A method for performing inverse memory compensation, comprising:

receiving a video bit stream;

identifying a transform matrix type selected from the group consisting of a half pixel matrix and a full pixel matrix;

if the transform matrix type is a half pixel matrix, the method includes,

applying a factorization technique to decode the bit stream corresponding to the half pixel matrix; and

if the transform matrix type is a full pixel matrix, the method includes,

applying an integer approximation technique to decode the bit stream corresponding to the full pixel matrix.

32. The method of claim 31, wherein the video bit stream is a low rate video bit stream.

33. The method of claim 31, wherein the method operation of applying the factorization technique to decode the bit stream corresponding to the half pixel matrix includes,

factoring the half pixel matrix into a sequence of sparse matrices, the sparse matrices including permutation matrices and diagonal matrices.

34. The method of claim 31, wherein the method operation of applying an integer approximation technique to decode the bit stream corresponding to the full pixel matrix includes,

approximating each element of the full pixel matrix with binary numbers.

35. The method of claim 34, wherein each element is rounded to a nearest power of two.

36. A method for decoding video data, comprising:

receiving a frame of video data within a compressed bit stream;

decoding a block of the frame into a transform domain representation in the compressed domain;

storing data associated with the transform domain representation in a hybrid data structure;

performing inverse motion compensation on the data associated with the transform domain representation in the compressed domain; the performing inverse motion compensation including,

determining a type of transform matrix associated with a portion of the frame of video data; and

applying a hybrid factorization and integer approximation technique to enhance inverse motion compensation.

37. The method of claim 36, wherein the compressed bit stream is associated with a standard selected from the group consisting of H.263, H.261 and Motion Pictures Expert Group.

38. The method of claim 36, wherein the hybrid data structure includes a fixed size array and a variable size overflow vector.

39. The method of claim 36, wherein the type of transform matrix is selected from the group consisting of a half pixel matrix and a full pixel matrix.

40. The method of claim 39, wherein the half pixel matrix is associated with a high motion region of an image and the full pixel matrix is assoc

iated with a minimal motion region of the image.

41. The method of claim 36, wherein the method operation of applying a hybrid factorization and integer approximation technique to enhance inverse motion compensation includes,

applying a factorization technique to matrices associated with blocks corresponding to high motion regions of the frame; and

applying an integer approximation technique to remaining blocks of the frame.

42. The method of claim 36, wherein the compressed bit stream is a low rate bit stream.

43. A computer readable media having program instructions for performing inverse motion compensation in a compressed domain, comprising:

program instructices for identifying a transform matrix;

program insiructions for determining if the transform matrix is one of a half pixel matrix and a full pixel matrix;

program instructions for applying a factorization technique to decode blocks of the bit stream corresponding to the half pixel matrix; and

program instructions for applying an integer approximation technique to decode blocks of the bit stream corresponding to the full pixel matrix.

44. The computer readable media of claim 43, wherein the program instructions for performing inverse motion compensation is executed in the compressed domain.

45. The computer readable media of claim 43, further including:

program instructions for extracting motion vector data, the motion vector data identifying the transform matrix as one of the half pixel matrix and the full pixel matrix.

46. The computer readable media of claim 43, further including:

program instructions for arranging non-zero transform coefficients associated with a coded block of a frame of data into a hybrid data structure

47. The computer readable media of claim 43, wherein the program instructions for applying an integer approximation technique to decode blocks of the bit stream corresponding to the full pixel matrix includes,

program instructions for approximating each element of the full pixel matrix with binary numbers.

48. The computer readable media of claim 43, wherein the program instructions for applying a factorization technique to decode blocks of the bit stream corresponding to the half pixel matrix includes,

program instructions for factoring the half pixel matrix into a sequence of sparse matrices, the sparse matrices including permutation matrices and diagonal matrices.

49. A circuit, comprising:

an integrated circuit chip configured to decode video data, the integrated circuit chip including,

circuitry for receiving a bit stream of data associated with a frame of video data;

circuitry for decoding the bit stream of data into a transform domain representation;

circuitry for identifying a type of transform matrix; and

circuitry for performing inverse motion compensation through a hybrid factorization and integer approximation technique.

50. The circuit of claim 49, wherein the integrated circuit chip further includes:

circuitry for arranging non-zero transform coefficients of the transform domain representation in a hybrid data structure.

51. The circuit of claim 49, wherein the bit stream is a low rate bit stream.

52. The circuit of claim 49, wherein the circuitry for performing invers

e motion compensation through a hybrid factorization and integer approxi mation technique is configured to apply a factorization technique to a h alf pixel transform matrix and an integer approximation technique to a f ull pixel transform matrix.

53. The circuit of claim 49, further including a memory in communication with the integrated circuit chip.

54. The circuit of claim 49, wherein the hybrid factorization and intege r approximation technique is applied to data in the compressed domain.

55. A video decoder, comprising:

a variable length decoder (VLD) configured to extract coefficient values and motion vector data from an incoming bit stream;

a dequantization block in communication with the VLD, the dequantization block configured to rescale the coefficient values;

a lower branch in communication with the dequantization block, the lower branch configured to decode error coefficients into a spatial domain; a nd

an upper branch in communication with the dequantization block, the uppe r branch configured to maintain an internal transform domain representat ion, the upper branch configured to generate a spatial domain output cap able of being added to the decoded error coefficients to reconstruct a c urrent block.

56. The video decoder of claim 55, wherein the video decoder is implemen ted in software.

57. The video decoder of claim 55, wherein the video decoder is implemen ted in hardware.

58. The video decoder of claim 55, wherein the incoming bit stream is a low rate bit stream.

59. The video decoder of claim 55, wherein the upper branch includes a f eedback loop, the feedback loop including a frame buffer, a motion compe

nsation block and a discrete cosine transform block.

60. The video decoder of claim 55, wherein the lower branch includes a r un length decode block and an inverse transform block.

61. The video decoder of claim 55, wherein inverse motion compensation o perations are performed in a compressed domain.

62. The video decoder of claim 55, wherein non-zero coefficients of the transform domain representation are arranged in a hybrid data structure in memory associated with the video decoder in order to reduce memory re quirements.

63. The video decoder of claim 62, wherein the hybrid data structure inc ludes a fixed size array and a variable size overflow vector

64. The video decoder of claim 61, wherein the inverse motion compensati on includes a hybrid factorization and integer approximation technique.

65. The video decoder of claim 64, wherein the hybrid factorization and integer approximation technique is configured to apply a factorization t echnique to a half pixel transform matrix and an integer approximation t echnique to a full pixel transform matrix.

3.Background of the Invention

Field of the Invention

This invention relates generally to digital video technology and more p articularly to a method and apparatus for implementing efficient memory compression methods and to a method and apparatus for implementing effic ient inverse motion compensation methods for a compressed domain video d ecoder.

Description of the Related Art

The access of video on mobile terminals, such as cellular phones and pe rsonal digital assistants, presents many challenges because of the limit

ations due to the nature of the mobile systems. For example, low-powere d, handheld devices are constrained under bandwidth, power, memory, and cost requirements. The video data received by these handheld devices ar e decoded through a video decoder. The video decoders associated with s uch terminals perform motion compensation in the spatial domain, i.e., d ecompressed domain. Video compression standards, such as H.263, H261 an d MPEG-1/2/4, use a motion-compensated discrete cosine transform (DCT) s cheme to encode videos at low bit rates. As used herein, low bit rates refer to bit rates less than about 64 kilobits per second. The DCT sche me uses motion estimation (ME) and motion compensation (MC) to remove te mporal redundancy and DCT to remove the remaining spatial redundancy.

Figure 1 is a schematic diagram of a video decoder for decoding video d ata and performing motion compensation in the spatial domain. Bit strea m 102 is received by decoder 100. Decoder 100 includes variable length decoder (VLD) stage 104, run length decoder (RLD) stage 106, Dequantizat ion (DQ) stage 108, inverse discrete cosine transform (IDCT) stage 110, motion compensation (MC) stage 112 and memory (MEM) 114, also referred t o as a frame buffer. The first four stages (VLD 104, RLD 106, DQ 108, a nd IDCT 110) decode the compressed bit stream back into the pixel domain . For an intracoded block, the output of the first four stages, 104, 10 6, 108 and 110, is used directly to reconstruct the block in the current frame. For an intercoded block, the output represents the prediction e rror and is added to the prediction formed from the previous frame to re construct the block in the current frame. Accordingly, the current fram e is reconstructed on a block by block basis. Finally, the current fram e is sent to the output of the decoder, i.e., display 116, and is also s tored in frame buffer (MEM) 114.

MEM 114 stores the previously decoded picture required by motion compen sation 112. The size of MEM 114 must scale with the incoming picture fo

rmat. For example, H.263 supports five standardized picture formats: (1)sub-quarter common intermediate format, (sub QCIF), (2) quarter common intermediate format (QCIF), (3) common intermediate format (CIF), (4) 4CIF, and (5) 16CIF. Each format defines the width and height of the picture as well as its aspect ratio. As is generally known, pictures are coded as a single luminance component and two color difference components (Y,Cr,Cb). The components are sampled in a 4:2:0 configuration, and each component has a resolution of 8 bits/pixel. For example, the video decoder of Figure 1 must allocate approximately 200 kilobytes of memory for MEM 114 while decoding a H.263 bit stream with CIF format. Furthermore, when multiple bit streams are being decoded at once, as required by video conferencing systems, the demands for memory become excessive.

MEM 114 is the single greatest source of memory usage in video decoder 100. In order to reduce memory usage, one approach might be to reduce the resolution of the color components for the incoming bit stream. For example, if the color display depth on the mobile terminal can only show 65,536 colors, then it is possible to reduce the resolution of the color components (Y,Cr,Cb) from 24 bits/pixel down to 16 bits/pixel. While this technique can potentially reduce memory usage by 30%, it is a display dependent solution that must be hardwired in the video decoder. Also, this technique does not scale easily with changing peak signal-to-noise ratio (PSNR) requirements, therefore, this approach is not flexible.

Operating on the data in the spatial domain requires increased memory capacity as compared to compressed domain processing. In the spatial domain, the motion compensation is readily calculated and applied to successive frames of an image. However, when operating in the compressed domain motion compensation is not as straightforward as a motion vector pointing back to a previous frame since the error values are no longer spatial values, i.e., the error values are not pixel values when operating in

the compressed domain. Additionally, methods capable of efficiently handling compressed domain data are not available. Prior art approaches have focused mainly on transcoding, scaling and sharpening compressed domain applications. Additionally, inverse compensation applications for the compressed domain tend to give poor peak signal to noise ratio (PSNR) performance and at the same time have an unacceptably slow response time in terms of the amount of frames per second that can be displayed.

As a result, there is a need to solve the problems of the prior art to provide a method and apparatus that minimizes the demands on memory for decoding low bit rate video data and to provide a method and apparatus to enable fast and efficient inverse motion compensation for a compressed domain video recorder.

Summary of the Invention

Broadly speaking, the present invention fills at least one aspect of these needs by providing a video decoder configured to minimize the memory requirements through the use of a hybrid data structure. It should be appreciated that this aspect of the present invention can be implemented in numerous ways, including as a method, a system, computer readable media or a device. Several inventive embodiments of this aspect of the present invention are described below.

In one embodiment, a method for reducing the memory requirements for decoding a bit stream is provided. The method initiates with receiving a video bit stream. Then, a frame of the bit stream is decoded into a transform (e.g., a discrete cosine transform (DCT)) domain representation. Next, non-zero coefficients of the transform domain representation are identified. Then, a hybrid data structure is assembled. The hybrid data structure includes a fixed size array and a variable size overflow vec

tor. Next, the non-zero coefficients of the transform domain representation are inserted into the hybrid data structure.

In another embodiment, a method for decoding video data is provided. The method initiates with receiving a frame of video data within a compressed bit stream. Then, a block of the frame is decoded into a transform (e.g., DCT) domain representation in the compressed domain. Next, a hybrid data structure is defined. Then, data associated with the transform domain representation is stored in the hybrid data structure. Next, inverse motion compensation is performed on the data associated with the transform domain representation in the compressed domain. After performing the inverse motion compensation on the data, the data is decompressed for display.

In yet another embodiment, computer readable media having program instructions for rearranging low rate bit stream data for storage into a hybrid data structure is provided. The computer readable media includes program instructions for identifying non-zero transform (e.g., DCT) coefficients associated with a coded block of a frame of data. Program instructions for arranging the non-zero transform coefficients into a fixed size array are included. Program instructions for determining if a quantity of the non-zero transform coefficients exceed a capacity of the fixed size array are provided. Program instructions for storing the non-zero transform coefficients exceeding the capacity of the fixed size array in a variable size overflow vector and program instructions for translating the non-zero transform coefficients from a compressed domain to a spatial domain are included.

In still yet another embodiment, a circuit is provided. The circuit includes a video decoder integrated circuit chip. The video decoder integrated circuit chip includes circuitry for receiving a bit stream of data associated with a frame of video data. Circuitry for decoding the bit

stream of data into a transform (e.g., DCT) domain representation is included in the video decoder. Circuitry for arranging non-zero transform coefficients of the transform domain representation in a hybrid data structure in a memory associated with the video decoder is provided. Circuitry for decompressing the non zero transform coefficients of the transform domain representation for display is also provided.

In another embodiment, a device configured to display an image is provided. The device includes a central processing unit (CPU), a random access memory (RAM), and a display screen configured to present an image. Decoder circuitry configured to transform a video bit stream into a transform (e.g., DCT) domain representation is included. The decoder circuitry is capable of arranging non-zero transform coefficients of the transform domain representation in a hybrid data structure in a memory associated with the decoder circuitry. The decoder circuitry includes circuitry for selectively applying a hybrid factorization/integer approximation technique during inverse motion compensation. A bus in communication with the CPU, the RAM, the display screen and the decoder circuitry is also included.

Broadly speaking, the present invention fills at least another aspect of these needs by providing a video decoder capable of performing inverse motion compensation in the compressed domain while reducing memory requirements and provide acceptable video quality. It should be appreciated that this aspect of the present invention can be implemented in numerous ways, including as a method, a system, computer readable media or a device. Several inventive embodiments of this aspect of the present invention are described below.

In one embodiment, a method for performing inverse memory compensation is provided. The method initiates with receiving a video bit stream. Then, a transform matrix type is identified. The transform matrix type i

s either a half pixel matrix or a full pixel matrix. If the transform matrix type is a half pixel matrix, then the method includes applying a factorization technique to decode the bit stream corresponding to the half pixel matrix. If the transform matrix type is a full pixel matrix, then the method includes applying an integer approximation technique to decode the bit stream corresponding to the full pixel matrix.

In another embodiment, a method for decoding video data is provided. The method initiates with receiving a frame of video data within a compressed bit stream. Then, a block of the frame is decoded into a transform (e.g., a discrete cosine transform (DCT)) domain representation in the compressed domain. Next, data associated with the transform domain representation is stored in a hybrid data structure. Then, inverse motion compensation is performed on the data associated with the transform domain representation in the compressed domain. Determining a type of transform matrix associated with a portion of the frame of video data, and applying a hybrid factorization and integer approximation technique to enhance inverse motion compensation are included in performing the inverse motion compensation.

[0017] In yet another embodiment, a computer readable media having program instructions for performing inverse motion compensation in a compressed domain is provided. The computer readable media includes program instructions for identifying a transform matrix. Program instructions for determining if the transform matrix is either a half pixel matrix or a full pixel matrix are included. Program instructions for applying a factorization technique to decode blocks of the bit stream corresponding to the half pixel matrix and program instructions for applying an integer approximation technique to decode blocks of the bit stream corresponding to the full pixel matrix are included.

In still yet another embodiment, a circuit is provided. The circuit in

cludes an integrated circuit chip configured to decode video data. The integrated circuit chip includes circuitry for receiving a bit stream of data associated with a frame of video data. Circuitry for decoding the bit stream of data into a transform (e.g., DCT) domain representation is included on the integrated circuit chip. Circuitry for identifying a type of transform matrix and circuitry for performing inverse motion compensation through a hybrid factorization and integer approximation technique are provided on the integrated circuit chip.

In another embodiment, a video decoder is provided. The video decoder includes a variable length decoder (VLD) configured to extract coefficient values and motion vector data from an incoming bit stream. A dequantization block in communication with the VLD is included. The dequantization block is configured to rescale the coefficient values. A lower branch in communication with the dequantization block is provided. The lower branch is configured to decode error coefficients into the spatial domain. An upper branch in communication with the dequantization block is included. The upper branch is configured to maintain an internal transform (e.g., DCT) domain representation. The upper branch is further configured to generate a spatial domain output capable of being added to the decoded error coefficients to reconstruct a current block.

Other aspects and advantages of the invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, illustrating by way of example the principles of the invention.

Detailed Description of the Preferred Embodiments

An invention is described for a system, apparatus and method for minimizing memory capacity for compressed domain video decoding. It will be a

pparent, however, to one skilled in the art, in view of the following de scription, that the present invention may be practiced without some or a ll of these specific details. In other instances, well known process op erations have not been described in detail in order not to unnecessarily obscure the present invention. Figure 1 is described in the "Backgroun d of the Invention" section. The term about as used to herein refers to +/- 10% of the referenced value.

The embodiments described herein provide data structures that enable th e reduction of the memory used while decoding video data in the compress ed domain. In one embodiment, the video decoding pipeline is rearranged such that the current frame is stored, and the inverse motion compensat ion is performed, in the frequency domain, i.e., compressed domain. Hyb rid data structures allow for the manipulation of the data in the compre ssed domain without computational cost or any significant loss of data. In one embodiment, the hybrid data structures take advantage of the fac t that there are only a small number of non-zero discrete cosine transfo rm (DCT) coefficients within a coded block. Thus, only the non-zero DCT coefficients of the entire frame are stored, thereby reducing the memor y requirements. As will be explained in more detail below, the hybrid d ata structure includes a fixed size array and a variable size overflow v ector. The variable size overflow vector stores the non-zero DCT coeffi cients of the coded blocks that exceed the capacity of the fixed size ar ray.

Figure 2 is a schematic diagram of a video decoder arranged such that i nverse motion compensation is performed in the compressed domain in acco rdance with one embodiment of the invention. Here, bit stream 122 is re ceived by video decoder 120. The first two stages variable length decod er (VLD) stage 124 and dequantization (DQ) stage 126, decode the compres sed bit stream into a DCT domain representation. The DCT domain represe

ntation is stored in memory (MEM) 130, also referred to as a frame buffe r, for use in motion compensation (MC) stage 128. Run length decoder (R LD) stage 132 and inverse DCT (IDCT) stage 134 is performed after the mo tion compensation feedback loop which contains MC 128 and MEM 130. Thus , the internal representation of the block being decoded is kept in the compressed domain. There are only a small number of nonzero DCT coeffic ients within a coded block. therefore, this characteristic can be exploi ted by developing data structures for MEM 130 that store only the nonzer o DCT coefficients of each block in the frame. As will be shown in more detail below, the memory compression enabled through the hybrid data st ructures can reduce memory usage by 50% without any loss in video qualit y. Since the human visual system is more sensitive to the lower order D CT coefficients than the higher order DCT coefficients, thresholding sch emes that filter out higher order DCT coefficients and tradeoff memory u sage versus changing power or peak signal to noise ratio (PSNR) requirem ents are developed as described below.

Accordingly, a complete compressed domain video decoding pipeline that is optimized for both fast and memory efficient decoding is described he rein. In one embodiment, TELENOR's video decoder, which is a public dom ain H.263 compliant decoder, is used for the testing referred to herein.

It should be appreciated that while some of the embodiments described below refer to a H.263 bit stream, the embodiments are not limited to op erating on a H.263 bit stream. That is, any DCT based compressed bit st ream having video data, e.g., Motion Picture Expert Group (MPEG) 1/2/4, H.261, etc. may be employed. A number of fast inverse motion compensati on algorithms for the discrete cosine transform (DCT) domain representat ion enable the efficient processing in the compressed domain. It should be appreciated that memory compression methods that store the nonzero D CT coefficients within a coded block allow for the reduction in memory r

equirements due to the compressed domain processing. Additionally, perf ormance of the video decoder using compressed domain processing with the inverse motion compensation techniques and memory compression described herein is evaluated along three dimensions: computational complexity, m emory efficiency, and PSNR, to show the various performance tradeoffs in optimizing for both speed and memory.

Figure 3 is a schematic diagram illustrating inverse motion compensation as peformed in the spatial domain. Here, a prediction of the current block is performed from motion compensated blocks in the reference frame. The current 8x8 spatial block, f_k 142, of current frame 140 is derived from four reference blocks f_1', f_2', f_3', and f_4', 144-1 through 144-4, respectively, in reference frame 146. The reference blocks are selected by calculating the displacement of f_k by the motion vector $(\Delta x,\Delta y)$ and choosing those blocks that the motion vector intersects in the reference frame. For $(\Delta x>0,\Delta y>0)$, f_k is displaced to the right and down. From the overlap of f_k with f_1', we can determine the overlap parameters (w,h) and also the parameters $(8-w,h)$, $(w,8-h)$, and $(8-w,8-h)$ with the neighboring blocks.

$$f_k=\sum_{i=1}^{4}c_{i1}f_i'c_{i2} \qquad (2)$$

Since each block can be represented as an 8x8 matrix, the reconstruction of matrix f_t can be described as the summation of windowed and shifted matrices $f_1', \ldots, f_4'$. In equation (Eq.) (2), the matrices c_{ij}, i = 1,...,4, j = 1,2, perform the windowing and shifting operations on f_i'. The matrices c_{ij} are sparse 8x8 matrices of zeroes and ones. Also, c_{ij} is a function of the overlap parameters (w,h) and is defined as

$$c_{11} = c_{21} = U_h = \begin{pmatrix} 0 & I_h \\ 0 & 0 \end{pmatrix}, \tag{3}$$

$$c_{12} = c_{32} = L_w = \begin{pmatrix} 0 & 0 \\ I_w & 0 \end{pmatrix}, \tag{4}$$

where I_h and I_w are identity matrices of dimension h x h and w x w, respectively. Similarly,

$$c_{31} = c_{41} = L_{8-h}, \tag{5}$$

$$c_{22} = c_{42} = L_{8-w}. \tag{6}$$

The inverse motion compensation in the DCT-domain reconstructs intracoded blocks from motion compensated intercoded blocks. The concept is similar to the spatial domain except that all coefficients are kept in the DCT-domain, i.e. reconstruct F_k, the DCT of f_k, directly from $F_1', \ldots, F_4'$, the DCT of $f_1', \ldots, f_4'$.

S is defined as a matrix that contains the 8x8 basis vectors for a two-dimensional DCT. Using the unitary property of the DCT transform, $S^tS = I$, it can be demonstrated that Eq. (2) is equivalent to

$$f_k = \sum_{i=1}^{4} c_{i1} S^t S f_i' S^t S c_{i2}. \qquad (7)$$

Premultiplying both sides of Eq. (7) by S, and postmultiplying by S^t, results in:

$$F_k = \sum_{i=1}^{4} C_{i1} F_i' C_{i2}, \qquad (8)$$

where C_{ij} is the DCT of c_{ij}. Eq. (8) calculates F_k as a summation of pre- and post-multiplied terms $F_1', \ldots, F_4'$. The matrix C_{ij} is a single composite matrix that contains the sequence of transformations: inverse DCT, windowing, shifting, and forward DCT. Thus, Eq. (8) describes a method to calculate F_k directly from $F_1', \ldots, F_4'$ using only matrix multiplications. These matrix multiplications operate in the DCT-domain without having to explicitly transform between the spatial and frequency domains. However, the matrix multiplications described are unacceptably slow. In turn, only about 5 frames per second can be displayed which results in a poor quality display. The DCT-domain inverse motion compensation algorithms described below focus on reducing the computational complexity of these matrix multiplications as the matrix multiplications become a bottleneck causing unacceptable delays.

Low bit rate video, i.e., video data having bit rates less than about 64 kilobits per second, is targeted for applications such as wireless video on cellular phones, personal digital assistants PDAs, and other handheld or battery operated devices, as well as being used for video conferencing applications. The H.263 standard is an exemplary standard that specifies the bit stream syntax and algorithms for video coding at low bit rates. The algorithms include transform coding, motion estimation/compensation, coefficient quantization, and run-length coding. Besides the baseline specification, version 2 of the standard also supports sixteen negotiable options that improve coding performance and provide error resilience.

Video encoded at low bit rates can become visibly distorted, especially those classified with high action, i.e., active motion blocks. As ment

ioned above, the embodiments described herein refer to the H.263 standar d, however any suitable video codec standard can be employed with the em bodiments. Some of the characteristics of the features of the H.263 sta ndard are discussed below for informational purposes and are not meant t o limit the invention for use with the H.263 standard. One characterist ic of the H.263 standard is the absence of the group of pictures (GOP) a nd higher layers in the H.263 standard. Where baseline encoded sequence s composed of just a single intraframe (I frame) followed by a long seq uence of interframes (P frames), the long sequence of P frames provides g reater compression ratios since the temporal redundancy is removed betwe en consecutive frames. However, motion estimation/ motion compensation (ME/MC) also creates a temporal dependency such that errors generated du ring the lossy coding process will accumulate during the decoding proces s. The lack of I frames prevents the decoder from breaking this accumul ation of errors. The H.263 standard has a forced update mechanism such that the encoder must encode a macroblock as an intrablock at least once every 132 times during the encoding process. Figure 4 is a graph illus trating the effectiveness of the forced update mechanism. As illustrate d in Figure 4, the PSNR of the video fluctuates randomly but does not dr ift in any one direction for frames later in the sequence.

Figure 5 is a schematic diagram illustrating the determination of half pixel values in the H.263 standard. As is well known, the H.263 standar d uses half pixel interpolation for motion compensation. In the standar d, half pixel interpolation is indicated by motion vectors with 0.5 reso lution (i.e. <7.5, 4.5>). The encoder can specify interpolation in the horizontal direction only, vertical direction only, or both horizontal a nd vertical directions. As illustrated by Figure 5, half pixel values a re found by bilinear interpolation of integer pixel positions surroundin g the half pixel position. Pixel position A 150-1, pixel position B 150

-2, pixel position C 150-3, and pixel position D 150-4, represent integer pixel positions, while position e 152-1, position f 152-2, and position g 152-3 represent half pixel positions. Interpolations in the horizontal direction may be represented as e=(A+B+1)>>1 and interpolations in the vertical direction may be represented as f=(A+C+1)>>1. Interpolations in the horizontal and vertical directions may be represented as g=(A+B+C+D+2)>>2

Figures 6A and 6B are schematic diagrams of a baseline spatial video decoder and a compressed domain video decoder, respectively. The block diagram of Figure 6B rearranges some of the functional blocks of the spatial domain video decoder of Figure 6A. In particular, RLD 132 and IDCT 134 are moved after MC 128 feedback loop. This arrangement keeps the internal representation of the video in the compressed domain. The arrangement of Figure 6B allows for the insertion of compressed domain post processing modules right after MC 128 feedback loop. It should be appreciated that certain video manipulations, such as compositing, scaling, and deblocking, to name a few, are faster in the compressed domain over their spatial domain counterparts. However, from the video codec point of view, a spatial encoder is not perfectly matched to a compressed domain decoder. As shown in Figure 6B, the compressed domain video decoder differs from that of the spatial domain video decoder of Figure 6A at several points along the decoding pipeline. More than just a rearrangement of blocks, the points of difference represent nonlinear operations, such as clipping and rounding. These points of nonlinearity generate video with differing PSNR measurements between the two domains.

The nonlinear points are labeled as (i), (ii), (iii), (iv), and (v). In the spatial decoder of Figure 6A, IDCT block 134 transforms the incoming 8x8 block from the frequency domain to the spatial domain. The spatial domain values represent either pixel values or prediction error values for the color channels (Y,Cr,Cb). At point (i) of Figure 6A, the spatial values are clipped to the range ($-255 \leq x \leq 256$). Note that there is no equivalent clipping operation at this stage for the DCT coefficients in Figure 6B. The second point of difference occurs during motion compensation. MC block 128 in Figure 6A returns the pixel values from MEM 130 referenced by the current motion vector. At point (ii) of Figure 6A, half-pixel (HP) interpolation 160, if specified, averages the neighboring pixel values and rounds the result to the nearest positive integer. At point (iv) of Figure 6B, half-pixel (HP) interpolation 160 operates directly on DCT coefficients and rounds the result to the nearest positive or negative integer. Another point of difference occurs after the addition of the prediction error to the prediction value. At point (iii) of Figure 6A, the sum represents pixel values, which are clipped at block 162b to the range ($0 \leq x \leq 255$). Note that in Figure 6B similar clipping of pixel values is moved from the motion compensation feedback loop to the last stage of the decoding pipeline at block 162 (point v).

One skilled in the art will appreciate that, MEM 130 is a frame buffer that stores the previous frame for motion compensation. For the spatial domain decoder, the frame buffer allocates enough memory to store the

(Y,Cr,Cb) values for the incoming frame size. For example, CIF video sampled at 4:2:0 requires about 200 kilobytes of memory. As MEM 130 is the single greatest source of memory usage in the video decoder, a hybrid data structure and inverse motion compensation methods defined herein allow for the reduction of MEM usage for a compressed domain decoding pipeline. In one embodiment, two to three times memory compression, without any significant loss in the quality of the decoded video, is achieved.

Figure 7 is a block diagram illustrating the block transformations during the video encoding and decoding process in accordance with one embodiment of the invention. The sequence of transformations above dotted line 170 describes the spatial compression methods used by the video encoder for a block in an I-frame, or a block in a P-frame after motion compensation/motion estimation. Pixel block 172 is a full 8x8 matrix. At this point, any compression or truncation in the spatial domain directly affects the perceived quality of the reconstructed block. After the DCT transform, however, transformed matrix 174 is compact with the larger terms at low frequencies. The quantization step further compacts the block by reducing to zero the smaller terms at high frequencies in block 176. The zigzag scan highlighted in block 176 orders the DCT coefficients from low to high frequency. The runlength encoding discards the zero coefficients and represents only the nonzero DCT coefficients in a compact list of two-valued elements, e.g., run and level, in runlength representation 178. Thus, memory compression in the DCT domain can be achieved by developing efficient data structures and methods that store and access runlength representations of the nonzero DCT coefficients.

In one embodiment, a semi-compressed (SC) representation is one such memory efficient runlength representation. The runlength representation of the nonzero DCT coefficients similar to runlength representations 178 and 180 of Figure 7. However, there are two modifications. Each two-va

lued element (run, level) is described by a composite 16-bit value of th e form:

RL = binary 'rrrrllllllllllll' (9)

The 12 least significant bits ('llllllllllll') define the value of the d equantized DCT coefficient from block 184, which were derived from quant ized block 182. It should be appreciated that block 184 is an example o f a DCT domain representation. It will be apparent to one skilled in th e art that the value of the DCT coefficients can range from -2048 to 204 7. Block 186 of Figure 7 is a reconstructed block of block 172 after an IDCT operation is performed on block 184. The four most significant bi ts ('rrrr') define the value of the run. The run represents the position of the nonzero DCT coefficient relative to the position of the last non zero DCT coefficient according to the zigzag scan in an 8x8 block. Sinc e the run of a nonzero coefficient may exceed 15, an escape sequence is defined to split the run into smaller units. The escape sequence RL='F0' is defined to represent a run of 15 zero coefficients followed by a coe fficient of zero amplitude.

In order to reduce the memory requirements, data structures to store an d access the SC representation must be developed. The following data st ructures were considered: array, linked list, vector, and hybrid. In de veloping these structures, a balance between the need for memory compres sion and the need to maintain low computational complexity is taken into consideration and discussed further with reference to Table 1 below. W hile the SC representation provides the targeted memory compression, cer tain data structures will greatly increase the computational complexity of the decoder in three areas. First, by employing the two-byte represe ntation, the values of the (run, level) are not immediately available. Functions to pack and unpack the bits are needed for every access and mo dification to these values. Secondly, motion compensation is now compli

cated by the compact runlength representation. Thirdly, sort and merge operations are needed to add the prediction error to the prediction.

Figure 8 is a schematic diagram illustrating the use of a separate index to find the starting position of each 8x8 block in the runlength representation. If a single list 190, also referred to as vector, is used to store the runlength representation for all 8x8 blocks 192-1 through 192-4 in a frame, then access to a particular DCT block during motion compensation requires a separate index to lookup its start position, which complicates the motion compensation.

Figures 9A and 9B illustrate the sort and merge operations needed to add the prediction error to the prediction for an array-based data structure and a list data structure, respectively. In Figure 9A an array-based data structure requires only the addition of values at corresponding array indices. However, the array based data structure does not offer memory compression advantages. In Figure 9B, a list (or vector) data structure requires additional sort and merge operations. That is, the merge algorithm requires insertion and deletion functions, which can be very expensive in terms of computational complexity for data structures such as vectors. More particularly, if indices are equal then the DCT coefficients can be added or subtracted, e.g., (0,20) + (0,620)=(0,640). DCT coefficients are inserted if index in error precedes that in prediction. e.g., insert (0,-3). DCT coefficients are deleted if addition of DCT values equals 0, e.g., (1,13) + (4,-13)=(1,0).

Table 1 compares the memory compression ratios and computational costs for various data structures. While array-based data structures incur no additional computational costs besides the 64 additions needed for the prediction updates, an array of DCT coefficients provides no memory compression over the array of pixels since each DCT coefficient needs two-bytes instead of one for storage. A linked list or vector of semi-compres

sed (SC) representation provides up to 2.5 times memory compression over the array of pixels. However, neither solution is optimal since the insertion/deletion cost for a vector is expensive, especially insertions and deletions in the middle of the vector and the memory overhead for a linked list is expensive, as internal pointers are created for every element in the list.

TABLE 1

Data Structure	Memory Size (kilobytes)	Insertion/Deletion Cost	Memory Overhead	Compression Ratio
Array of Pixels	152	None	None	None
Array of DCT	304	None	None	None
Vector of SC	60	Expensive	Minimal	2.5:1
Linked List of SC	60+overhead	Moderate	Expensive	2.5:1(w/o overhead)
Hybrid of SC	70	Moderate	Minimal	2.2:1

A hybrid data structure for the SC representation provides the optimum balancing of the competing interests of Table 1. The hybrid data structure is developed to take advantage of the low computational cost of the array structure of Figure 9A and the high compression ratio of the vector structure of Figure 9B. The hybrid data structure consists of a fixed-size array that holds a fixed number of DCT coefficients per block and a variable-size overflow vector that stores the DCT coefficients of those blocks that exceed the fixed size array allocation. It should be appreciated that the fixed size array can be configured to hold any suitable number of DCT coefficients per block, wherein the number of DCT coefficients is less than 64. Of course, as the fixed size array becomes greater the amount of memory compression is decreased. In one embodiment, the fixed size array is configured to hold 8 DCT coefficients per block. Figure 10 is a schematic diagram of a hybrid data structure including an array structure and a vector structure to allow for memory compression and computational efficiency in accordance with one embodiment of the i

nvention. DCT blocks 200-1, 200-2 and 200-n include zero DCT coefficien ts and non-zero DCT coefficients. It should be appreciated that DCT blo cks 200-1 through 200-n represent the DCT domain representation as discu ssed above with reference to Figure 2. In addition, blocks 200-1 throug h 200-n are associated with blocks of a frame of video data, e.g., block 184 of Figure 7. The non-zero DCT coefficients for each of blocks 200- 1 through 200-n are identified and inserted into fixed size array 202 da ta structure. Fixed size array 202 includes fixed size blocks 204-1 thr ough 204-n. In one embodiment, each block 204-1 through 204-n is sized to store 8 DCT coefficients in an 8x1 data structure. It should be appr eciated that, the invention is not limited to blocks configured to store 8 DCT coefficients as any suitable size may be used. As stated above, a s the capacity of the blocks increases the amount of memory compression decreases.

Still referring to Figure 10, where there are more than 8 non-zero coef ficients in any of DCT blocks 200-1 through 200-n, the non-zero DCT coef ficients exceeding the capacity of respective fixed size blocks 204-1 th rough 204-n are placed in overflow vector 206. Overflow vector 206 is c onfigured as a variable size overflow vector, i.e. the overflow vector i s dynamic. For example, block 200-1 includes 9 non-zero DCT coefficient s A1-A9. Here, DCT coefficients A1-A8 are copied to fixed size block 20 4-1, while DCT coefficient A9 is copied to overflow vector 206. Block 2 00-2 includes 10 non-zero DCT coefficients B1-B10. Here, DCT coefficien ts B1-B8 are copied to fixed size block 204-2, while DCT coefficients B9 and B10 are copied to overflow vector 206 and so on for each block of t he frame. Index table 208 contains entries which identify corresponding fixed size blocks 204-1 through 204-n for the entries in overflow vecto r 206. The size of the index table is negligible as each entry is 1 byt e. Accordingly, for a frame of data corresponding to DCT blocks 200-1 t

hrough 200-n, data from fixed size array 202 and overflow vector 206 are combined to produce image 210. It should be appreciated that the savings in memory is substantial. That is, DCT blocks 200-1 through 200-n are reduced from 64 zero and non-zero coefficients to 8 non-zero coefficients, or less, stored in fixed size blocks 204-1 through 204-n in most instances. Of course, more or less non-zero coefficients may be provided, wherein the non-zero coefficients in excess of 8 are stored in overflow vector 206.

Figures 11A through 11C are graphs illustrating the factors evaluated in determining the capacity of the fixed size blocks of the fixed size array and the overflow vector of the hybrid data structure in accordance with one embodiment of the invention. In Figure 11A, the average number of non-zero DCT coefficients per luminance block for two typical CIF sequences is depicted by lines 220 and 222. The number of non-zero DCT coefficients per block ranges from three to seven. That is, of the 64 coefficients, only 2-7 coefficients are non-zero coefficients on average. Using the information from Figure 11A as a guide, Figure 11B illustrates that as the fixed-size array increases, the size of the overflow vector decreases, thereby minimizing the insertion and deletion costs of the vector. Here line 220-1 corresponds to the CIF sequence of line 220 of Figure 11A, while line 222-1 corresponds to the CIF sequence of line 222 of Figure 11A. One skilled in the art will appreciate that as the fixed size array increases in terms of capacity, the memory compression decreases. Additionally, Figure 11C illustrates that the load factor of the array decreases as well, indicating that much of the array remains empty.

In one embodiment, a fixed-size array that holds 8 DCT coefficients per block is chosen. Here again, line 220-2 corresponds to the CIF sequence of line 220 of Figure 11A, while line 222-2 corresponds to the CIF sequence of line 222 of Figure 11A. This choice minimizes the size of the

overflow vector to about 200 DCT coefficients and maintains a load factor of between about 9% and about 15%. It will be apparent to one skilled in the art that the size of the fixed array is not limited to 8 coefficients per block and that any suitable number of coefficients per block may be chosen. Additionally, the individual blocks of the fixed size array may have any suitable configuration. For example, a block capable of holding 8 coefficients may be arranged as an 8 x 1 block, a 4 x 2 block, etc., while a block capable of holding 9 coefficients may be arranged as a 9 x 1 block, 3 x 3 block, etc.

Figure 12 is a flowchart of the method operations for reducing the memory requirements for decoding a bit stream in accordance with one embodiment of the invention. The method initiates with operation 230 where a video bit stream is received. In one embodiment, the bit stream is a low rate bit stream. For example, the video stream may be associated with a video coding standard such as H.263, Motion Pictures Expert Group (MPEG-1/2/4), H.261, Joint Photographic Expert Group (JPEG), etc. The method then proceeds to operation 232 where the frame of the bit stream is decoded into a discrete cosine transform (DCT) domain representation for each block of data associated with the frame. Here, the video is processed through the first two stages of a decoder, such as the decoder of Figures 2, 6B and 15. That is, the video data is processed through the variable length decoder stage and the dequantization stage to decode the compressed bit stream into a DCT domain representation. It should be appreciated that the DCT domain representation is in a compressed state format. The frame is decoded one block at a time. The method then moves to operation 234 where the non-zero coefficients of the DCT domain representation are identified. Here, out of the 64 DCT coefficients associated with the DCT domain representation for a block of data, relatively few of the 64 DCT coefficients are typically non-zero coefficients.

Still referring the Figure 12, the method then moves to operation 236 w here a hybrid data structure is assembled. The hybrid data structure in cludes a fixed size array and a variable size overflow vector. One exem plary hybrid data structure is the fixed size array that includes a plur ality of fixed size block and the variable size overflow vector with ref erence to Figure 10. The method then proceeds to operation 238 where th e non-zero coefficients of the DCT domain representation are inserted in to the hybrid data structure. As mentioned with reference to Figure 10, the non-zero coefficients for a DCT domain representation for a block o f video data are associated with a fixed size block in the fixed size ar ray. If the number of non-zero coefficients exceeds the capacity of the fixed size block associated with the block of video data, then the rema ining non-zero coefficients are stored in the variable size overflow vec tor. In one embodiment, an index table maps the data in the overflow ve ctor back to the appropriate fixed size block in the fixed size array. Thus, the memory requirements are reduced through the hybrid data struct ure and the storage of the non-zero coefficients. More particularly, th e memory requirements can be reduced by 50% without any loss of video qu ality.

It should be appreciated that the non-zero coefficients for each DCT do main representation associated with a frame of data are stored in the hy brid data structure. The stored data for the frame is then combined and decompressed for display. Once the next frame is decoded into a DCT do main representation to be stored in the hybrid data structure, the data in the hybrid data structure associated with the previous frame is flush ed, in one embodiment. As will be explained further below, inverse moti on compensation is performed on the stored data in the compressed domain . The inverse motion compensation uses integer approximation for full p ixel inverse motion compensation and factorization for half pixel invers

e motion compensation.

The main components in the spatial H.263 video decoder include runlength decoding, inverse DCT, and inverse motion compensation. Using a timing profiler, the performance of TELENOR'S H.263 video decoder on a 1.1 GHz Pentium 4 processor is measured for baseline data. Decoding a baseline video and ignoring system calls, the profiler measures the overall time it takes to decode 144 frames and details the timing characteristics of each component. Table 2 is a timing profile for the spatial H.263 video decoder and highlights the timing results for select functions.

Table 2

Function	Function Time (ms)	Hit Count
Picture Display	772	144
Inverse Motion Compensation	243	56336
Runlength Decoding	57	39830
Inverse DCT	3	42253

Table 3 is timing profile for the non-optimized compressed domain H.263 video decoder. One exemplary decoder pipeline configuration is the decoder with reference to Figure 2.

Table 3

Function	Function Time (ms)	Hit Count
Inverse Motion Compensation	9194	56336
Picture Display	1547	144
Runlength Decoding	32	39830
Inverse DCT	652	340197

As shown in Table 2, the spatial domain video decoder takes about 1.2 seconds to decode 144 frames. The majority of the time is spent in the PictureDisplay function, which converts the color values of each frame from YUV to RGB in order to display it on a suitable operating system, s

uch as WINDOWS™. Functions such as runlength decoding, inverse DCT, an d inverse motion compensation take about 25% of the total time required to decode the video. Inverse motion compensation is especially fast in the spatial domain. Here, full pixel motion compensation simply sets a pointer to a position in memory or a frame buffer and copies a block of data, while half pixel motion compensation sets a pointer in memory and interpolates values using the shift operator. In contrast, Table 3 high lights some of the timing results for a non-optimized compressed domain video decoder. The non-optimized compressed domain decoder takes about 13.67 seconds to decode the same 144 frames.

The main bottleneck for the compressed domain decoder is the inverse motion compensation function. As described in Eq (8) above, full-pixel inverse motion compensation in the compressed domain requires a sum of four (TM_i) terms, where TM_i is defined as pre- and post- multiplying the 8x8 matrix block F'_i with transform matrix C_y.

$$F_k = TM_1 + TM_2 + TM_3 + TM_4 \qquad (10)$$

$$\text{where } TM_i = C_{i1} F'_i C_{i2} \qquad (11)$$

Table 4 defines the full-pixel transform matrices C_{ij}. Here, S represent the 8x8 DCT matrices, and U_k and L_k are defined in Equations 3-6 above.

Table 4

Full-pixel transform matrix	Matrix definition
$C_{11} = C_{21}$	$SU_k S'$
$C_{31} = C_{41}$	$SL_{8-k} S'$
$C_{12} = C_{32}$	$SL_w S'$
$C_{22} = C_{42}$	$SU_{8-w} S'$

Each 8x8 matrix multiplication requires 512 multiplies and 448 additions. As is known matrix multiplication is computationally expensive. Table 5 compares the optimization schemes, such as matrix approximation, matrix factorization, sharedblock for macroblocks, and a hybrid scheme for a compressed domain video pipeline such as the pipeline with reference to Figures 2, 6B and 15. The compressed domain video decoding pipeline should decode at a rate of about 15-25 frames per second (fps) in order to provide acceptable quality for handheld devices that support video formats such as the common intermediate format where each frame of data contains 352 lines with 288 pixels per line.

Table 5

Optimization	Decode Time (s)	# Frames	FPS	Comments
Spatial domain	9.79	144	14.71	Original TELENOR H.263 video decoder.
Matrix-matrix	14.17	144	10.16	Full 8x8 matrix multiplications for TM.
Approximation	9.82	144	14.66	Good time but poor PSNR.
Factorization	12.95	144	11.12	Good PSNR but poor time.
Sharedblock	14.85	144	9.70	No improvement here.
Hybrid	9.83	144	14.65	Good time and good PSNR.

One enhancement to a compressed domain video decoding pipeline is to reduce the number of TM_i operations in Eq (10) by block alignment. For example, to decode 144 frames of a sequence and measure block alignment rates at 36.7% of all blocks. Figure 13 is a schematic diagram illustrating three examples of block alignment to reduce matrix multiplication. Block alignment case 240 where $(w=8,h=4)$, block alignment case 242 where $(w=4,h=8)$, and block alignment case 244 where $(w=8,h=8)$ are each illustrated. In each of these examples 240, 242, and 244, TM_i operations are eliminated when the overlap with a corresponding block is zero. However, it should be appreciated that, in the DCT domain (compressed domain), block alignment does not yield savings when half-pixel interpolation is specified. The equations for half-pixel inverse motion compensation in the compressed domain are given below. For the example of $(w=8,h=8)$, half-pixel interpolation still requires four TM_i operations as illustrated in equations 12 and 13. Table 6 is provided for informational purposes to define the half pixel transform matrices C_{hpij}.

$$F_{hpk} = TM_{hp1} + TM_{hp2} + TM_{hp3} + TM_{hp4} \quad (12)$$

$$TM_{hpi} = C_{hpi1} F'_i C_{hpi2} \quad (13)$$

Table 6

Half-pixel transform matrix	Horizontal interpolation	Vertical interpolation	Horizontal & vertical
$C_{hp11} = C_{hp21}$	$SU_h S'$	$S(U_h + U_{h+1})S'$	$S(U_h + U_{h+1})S'$
$C_{hp31} = C_{hp41}$	$SL_{8-h}S'$	$S(L_{8-h} + L_{9-h})S'$	$S(L_{8-h} + L_{9-h})S'$
$C_{hp12} = C_{hp32}$	$S(L_w + L_{w+1})S'$	$SL_w S'$	$S(L_w + L_{w+1})S'$
$C_{hp22} = C_{hp42}$	$S(U_{8-w} + U_{9-w})S'$	$SU_{8-w}S'$	$S(U_{8-w} + U_{9-w})S'$

It should be noted that even for a perfectly aligned DCT block, half-pixel interpolation creates an overlap of one with the neighboring blocks.

Figure 14 is a schematic diagram of a half pixel interpolation for a perfectly aligned DCT block. The half pixel interpolation creates overlapping into neighboring blocks by one pixel width and one pixel height.

Increasing the speed of processing in the compressed domain decoding pipeline may be accomplished by rearrangement of the functional blocks of the decoder of Figure 2. With reference to Tables 2 and 3, the processing time for the inverse DCT block is much less in the spatial domain (3 ms) than in the compressed domain (652 ms). In the spatial domain, inverse DCT is applied before the feedback loop to the intrablocks and the error coefficients. In particular, the intrablocks and error coefficients make up less than 15% of all the blocks in the video. The other 85% of the time the inverse DCT function is simply skipped. In the compressed domain, inverse DCT is applied at the last stage of the pipeline to 100% of the blocks in each frame of the video.

Figure 15 is a schematic diagram illustrating the rearrangement of the functional blocks of a compressed domain video decoder to enhance the processing of the video data in accordance with one embodiment of the invention. Here, the functional blocks are rearranged and the compressed domain pipeline is split at two points. The first split occurs after VLD 124 and DQ 126 at point (i) 252. In the upper branch, the pipeline keeps an internal DCT domain representation for memory compression 128. In the lower branch, the pipeline moves the RLD and IDCT up to the front to decode the error coefficients into the spatial domain. The second split occurs during motion compensation (MC) at point (ii) 254. During motion compensation, a spatial domain output may be generated according to equation (7). The output can be directly added to the error coefficients

to reconstruct the current block at point (iii) 256 to be presented on display 136. DCT block 250 is inserted in the feedback loop to maintain the internal DCT representation. The combination of RLD 132 and IDCT 134 at point (i) 252 and the DCT at point (ii) 254 requires less computation than the IDCT block at the last stage of the pipeline in Figure 2. Table 7 shows that the rearrangement with reference to Figure 15 generates a 20% speedup that can be combined in addition to other optimization schemes described herein.

Table 7

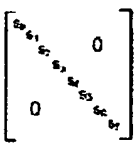
Function	Percentage of Blocks	Comments
IDCT in Figure 15 point (i)	15%	Intrablocks and error coefficients represent small fraction of all blocks.
DCT in Figure 15 point (ii)	63%	Non-aligned blocks require DCT, but aligned blocks are directly copied without DCT.
IDCT in Figure 2	100%	Applied to all blocks in DCT domain.

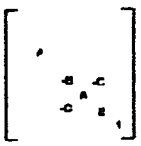
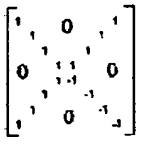
In one embodiment, the inverse motion compensation is accelerated by reducing the number of multiplies required by the basic TM operation in Eqs. (11,13). Instead of calculating full 8x8 matrix multiplications, the DCT matrix S is factored into a sequence of sparse matrices as illustrated in Eq. 14. The sparse matrices in Eq. (17) include permutation matrices ($A_1, A_2, A_3, A_4, A_5, A_6$) and diagonal matrices (D, M). Substituting this factorization into Eq. (15), we derive a fully factored expression for TM_i in Eq. (16), which requires less multiplies than the original Eqs. (11,13).

$$S = DA_1A_2A_3MA_4A_5A_6 \tag{14}$$

$$TM_i = Sc_{i1}S^t F'_i Sc_{i2}S^t \tag{15}$$

$$TM_i = (DA_1A_2A_3MA_4A_5A_6)c_{i1}(DA_1A_2A_3MA_4A_5A_6)^t F'_i \; (DA_1A_2A_3MA_4A_5A_6)c_{i2}(DA_1A_2A_3MA_4A_5A_6)^t \tag{16}$$

$D =$  $A_1 =$  $A_2 =$  $A_3 =$ 

$M =$  $A_4 =$  $A_5 =$  $A_6 =$ 

(17)

$$D = \text{diag}\{0.3536, 0.2549, 0.2706, 0.3007, 0.3536, 0.4500, 0.6533, 1.2814\} \tag{18}$$

$$A = 0.7071,\ B = 0.9239,\ C = 0.3827 \tag{19}$$

Thus, the matrix multiplication is replaced with matrix permutation. However, a fully factored expression for the term TM_i, as shown in Eq. (16), does not necessarily speed up inverse motion compensation. In essen

ce, multiplies have been traded for memory accesses, and too many memory accesses can actually slow down the decoding process. Therefore, the matrices are regrouped to strike a balance between these competing functionalities. Matrix S (= G_0G_1) is factored into two terms: $G_0=DA_1A_2A_3$, mixture of permutations and multiplications; and $G_1=MA_4A_5A_6$, mixture of permutations and additions. The fixed matrices J_i, K_i are defined and substituted into Eqs. (10 and 12) to form a factored expression for inverse motion compensation in Eq. (24):

$$J_h = c_{11}G_1^t = c_{21}G_1^t,\ J_w = G_1c_{12} = G_1c_{32} \tag{20}$$

$$K_h = c_{31}G_1^t = c_{41}G_1^t,\ K_w = G_1c_{22} = G_1c_{42} \tag{21}$$

Similarly for half-pixel interpolation:

$$J_h = c_{hp11}G_1^t = c_{hp21}G_1^t,\ J_w = G_1c_{hp12} = G_1c_{hp32} \tag{22}$$

$$K_h = c_{hp31}G_1^t = c_{hp41}G_1^t,\ K_w = G_1c_{hp22} = G_1c_{hp42} \tag{23}$$

$$F_k = S[J_hG_0^tF'_1G_0J_w + J_hG_0^tF'_2G_0K_w + K_hG_0^tF'_3G_0J_w + K_hG_0^tF'_4G_0K_w]S^t \tag{24}$$

Further speed enhancement may be obtained by implementing fast multiplication by the fixed matrices J_i, K_i. The fixed matrices contain repeated structures. For example, the matrix J_6 is defined as follows

$$J_6 = \begin{bmatrix} 1 & -1 & -a & 0 & b & a & c & 0 \\ 1 & 1 & -a & -1 & b & 0 & c & 0 \\ 1 & 1 & -a & -1 & -b & 0 & -c & 0 \\ 1 & -1 & -a & 0 & -b & -a & -c & 0 \\ 1 & -1 & a & 0 & c & -a & -b & 0 \\ 1 & 1 & a & 1 & c & 0 & -b & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

where a=0.7071, b=0.9239, and c=0.3827. To compute $u = J_6v$, where $u=[u_1,\ldots,u_8]$ and $v=[v_1,\ldots,v_8]$, a sequence of equations is calculated accordi

ng to the following steps:

$$y_1 = v_1 + v_2 \quad (25)$$

$$y_2 = v_1 - v_2 \quad (26)$$

$$y_3 = av_3 \quad (27)$$

$$y_4 = av_6 \quad (28)$$

$$y_5 = y_1 - y_3 \quad (29)$$

$$y_6 = y_5 - y_4 \quad (30)$$

$$y_7 = y_3 - y_4 \quad (31)$$

$$y_8 = y_3 + y_4 \quad (32)$$

$$y_9 = (b + c)(v_5 + v_7) \quad (33)$$

$$y_{10} = cv_5 \quad (34)$$

$$y_{11} = bv_7 \quad (35)$$

$$y_{12} = y_9 - y_{10} - y_{11} \quad (36)$$

$$y_{13} = y_{10} - y_{11} \quad (37)$$

$$u_1 = y_2 - y_7 + y_{12} \quad (38)$$

$$u_2 = y_6 + y_{12} \quad (39)$$

$$u_3 = y_6 - y_{12} \quad (40)$$

$$u_4 = y_2 - y_8 - y_{12} \quad (41)$$

$$u_5 = y_2 + y_7 + y_{13} \quad (42)$$

$$u_6 = y_1 + y_3 + u_4 + y_{13} - u_8 \quad (43)$$

$$u_7 = 0 \quad (44)$$

$$u_8 = 0 \quad (45)$$

Accordingly, the matrix-vector multiplication has been transformed into a sequence of equations. The above sequence of equations requires 5 multiplications and 21 additions. The matrix multiplication $J_h G_o^t F^t$ in Eq. (24) requires 104 multiplications and 164 additions. Thus, a 5 time reduction over the number of multiplies needed for matrix multiplication $C_{ft} F^t$ in Eq. (8) is achieved here. Additionally, no precision is lost during this matrix operation, which uses 32-bit floating point arithmetic. However, with reference to Table 5, factorization speeds up the compressed domain pipeline by only 9% over matrix-matrix. Consequently, the extra memory accesses slow the frame rate to below the target rate of about 15 to about 25 fps so that factorization alone will not suffice.

To further speedup the inverse motion compensation the multiplies required by the basic TM operation in Eqs. (11,13) are eliminated. The full-pixel and half-pixel matrices C_{ij} and C_{hpij} are approximated to binary numbers to the nearest power of 2^{-5}. By approximating these matrices with binary numbers, matrix multiplication can be performed by using basic integer operations, such as right-shift and add, to solve inverse motion compensation in Eqs. (10,12). For example, the full-pixel matrix C_{11} where h=1 is examined below. It should be appreciated that the other matrices are approximated in a similar fashion.

$$C_{11} = \begin{bmatrix} 0.12501651 & -0.17338332 & 0.16332089 & \cdots & -0.03447659 \\ 0.17340284 & -0.24048958 & 0.22653259 & \cdots & -0.04782041 \\ 0.16334190 & -0.22653624 & 0.21338904 & \cdots & -0.04504584 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0.03449197 & -0.04783635 & 0.04506013 & \cdots & -0.00951207 \end{bmatrix} \quad (46)$$

Where each element in the matrix is rounded to the nearest powers of 2, matrix (47) results:

$$\hat{C}_{11} \leftarrow \begin{bmatrix} 0.1250 & -0.1875 & 0.1875 & \cdots & -0.0625 \\ 0.1875 & -0.2500 & 0.2500 & \cdots & -0.0625 \\ 0.1875 & -0.2500 & 0.1875 & \cdots & -0.0625 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0.0625 & -0.0625 & 0.0625 & \cdots & 0 \end{bmatrix} \quad (47)$$

Since the DCT elements lie in the range of [-2048 to 2047], direct shifting of the DCT coefficients would drive most of the values to zero. In order to maintain precision in the intermediate results, we scale each DCT coefficient by 2^K throughout the decoding pipeline. This scaling factor is introduced during the quantization and dequantization steps so that no extra operations are incurred.

Furthermore, we implement fast matrix multiplication by grouping terms according to the sum of products rule (see Eqs. (48-50)).

$$u_1 = 0.1250v_1 - 0.1875v_2 + 0.1875v_3 - 0.1250v_4 + 0.1250v_5 - 0.1250v_6 + 0.0625v_7 - 0.0625v_8 \quad (48)$$

$$u_1 = (v_1 >> 3) - (v_2 >> 3) - (v_2 >> 4) + (v_3 >> 3) + (v_3 >> 4) - (v_4 >> 3) + (v_5 >> 3) - (v_6 >> 3) + (v_7 >> 4) - (v_8 >> 4) \quad (49)$$

$$u_1 = (v_1 - v_2 + v_3 - v_4 + v_5 - v_6) >> 3 + (-v_2 + v_3 + v_7 - v_8) >> 4 \quad (50)$$

The computation for $u = \hat{C}_{11} v$, where $u = \{u_1, \ldots, u_8\}$ and $v = \{v_1, \ldots, v_8\}$, may be calculated as:

$$u_1 = (v_1 - v_2 + v_3 - v_4 + v_5 - v_6) >> 3 + (-v_2 + v_3 + v_7 - v_8) >> 4 \quad (51)$$

$$u_2 = (v_3 - v_2) >> 2 + (v_1 - v_4 + v_5 - v_6 + v_7) >> 3 + (v_1 - v_4 + v_5 - v_8) >> 4 \quad (52)$$

$$u_3 = (v_1 + v_3 - v_4 + v_5 - v_6) >> 3 - (v_2 >> 2) + (v_1 + v_3 - v_4 + v_5 + v_7 - v_8) >> 4 \quad (53)$$

$$u_4 = (v_1 - v_2 + v_3 - v_4 + v_5 - v_6) >> 3 + (v_3 - v_2 - v_4 + v_7 - v_8) >> 4 \quad (54)$$

$$u_5 = (v_1 - v_2 + v_3 - v_4 + v_5 - v_6) >> 3 + (-v_2 + v_3 + v_7 - v_8) >> 4 \quad (55)$$

$$u_6 = (v_1 - v_2 + v_3 - v_4 + v_5) >> 3 + (v_7 - v_6) >> 4 \quad (56)$$

$$u_7 = (v_1 + v_3 - v_4 + v_5 - v_6 + v7) >> 4 + (v_2) >> 3 \quad (57)$$

$$u_8 = (v_1 - v_2 + v_3 - v_4 + v_5) >> 4 \quad (58)$$

The matrix approximation requires a total of 17 right-shifts and 57 adds. The matrix approximation $\hat{C}_{II}F'$ in Eq. (8) requires 136 right-shifts and 456 adds. Accordingly, a significant reduction in complexity over matrix multiplication is achieved with floating point precision. In fact, Table 5 shows that approximation techniques speed up the compressed domain pipeline by 31%, which is enough to achieve the target frame rate of about 15 fps. However, the PSNR for a sample video decreases and shows noticeable drift in areas of moderate motion.

A hybrid factorization/integer approximation for the transform matrix T M that is selectively applied based upon the video motion provides the d esired frame rate of between about 15 and about 25 fps, while maintainin g acceptable quality. As mentioned above, the integer approximation tec hnique reduces the complexity of the decoder but also reduces the PSNR o f the decoded video. At the same time, the factorization method maintai ns good PSNR but does not reduce the complexity of the decoder to meet t he desired frame rate. Through the integration of the low complexity of the integer approximation with the high precision of the factorization method a compressed domain video decoding pipeline for supporting a low rate video bit stream is obtained.

Two types of transform matrices have been discussed herein: TM_i, full pixel motion compensation illustrated in Eq. (11); and TM_{hpi}, half pixel motion compensation illustrated in Eq. (13). Full pixel motion compensation, using approximate matrices for TM_i, has only 28% of the computational complexity compared to that of using 8x8 floating point matrices. However, when applying the approximation techniques directly on the half pixel transform matrices, TM_{hpi}, it has been observed that half pixel motion compensation, using approximate matrices for TM_{hpi}, lowers the PSNR (see Table 8) and creates visible distortions in the decoded video. The errors are generated from two sources. First, the half pixel transform matrices TM_{hpi} are more sensitive to approximation techniques. With reference to Table 8, TM_{hpi} are composite matrices, composed of many more terms than TM_i. Secondly, as described above with reference to Figures 6A and 6B, the nonlinear processing during half pixel interpolation, combined with the errors generated by the approximation techniques, lead to an accumulation of errors that are especially visible in regions of moderate to high motion.

The selective application of the factorization method to the half pixel matrices addresses these errors. As discussed above, the factorization method maintains floating point precision so that the errors described can be minimized. For example, the factorization method reduces the matrix multiplication with TM_{hpi} into a sequence of equations similar to those described in Eqs. (25-45). These equations maintain 32-bit floating point precision so that no approximation errors are generated. Furthermore, the factorization methods decode the DCT block into the spatial domain during motion compensation so that the optimizations described with reference to Figure 15 may be combined with those described here. Table 5 shows that the hybrid method meets our target frame rate of 15 fps, while Table 8 illustrates that the PNSR of the hybrid method provides an acceptable PSNR.

Table 8

Video (128 kbps, QCIF, 15 fps)	Compressed Domain w/ Factor TM (PSNR_Y)	Compressed Domain w/ Hybrid TM (PSNR_Y)	Compressed Domain w/ Approximate TM (PSNR_Y)
Sample A	25.53	25.53	22.65
Sample B	22.47	19.57	18.75
Sample C	30.79	30.66	29.90
Sample D	33.29	33.25	28.93
Sample E	31.27	31.10	28.89

Figure 16 is a flowchart diagram of the method operations for performing inverse motion compensation in the compressed domain in accordance with one embodiment of the invention. The method initiates with operation 260 where a frame of video data within a compressed bit stream is received. In one embodiment, the bit stream is a low rate bit stream. For example, the bit stream may be associated with a known video coding standa

rd, such as MPEG 4, H.263, H.261, etc. The method then advances to operation 262 where a block of the frame of the bit stream is decoded into a discrete cosine transform (DCT) domain representation. Here, the video is processed through the first two stages of a decoder such as the decoder of Figures 2, 6B and 15. That is, the video data is processed through the variable length decoder stage and the dequantization stage to decode the compressed bit stream into a DCT domain representation. It should be appreciated that the DCT domain representation is in a compressed state format. The method then proceeds to operation 264 where the data associated with the DCT domain representation is stored in a hybrid data structure. A suitable hybrid data structure is the hybrid data structure discussed with reference to Figures 10 and 12. In one embodiment, the hybrid data structure reduces the memory requirements for a portable electronic device, e.g., cellular phone, PDA, web tablet, pocket personal computer, etc., having a display screen for presenting the video data.

Still referring to Figure 16, the method moves to operation 266 where inverse motion compensation is performed on the data associated with the DCT domain representation in the compressed domain. Here, the inverse motion compensation includes selectively applying a hybrid factorization/integer approximation technique described above with reference to Tables 5 and 8. The method then advances to decision operation 268 where the hybrid factorization/integer approximation identifies a type of transform matrix associated with the block of video data being processed. In one embodiment, the type of transform matrix is detected through information in a bit set of the bit stream being decoded. If the transform matrix is a half pixel matrix then the method proceeds to operation 270 where a factorization technique is applied to decode the bit stream. In one embodiment, the factorization technique reduces matrix multiplication in

to a series of equations as described above with reference to equation 2 5-45. That is, matrix multiplication is replaced with matrix permutatio n. If the transform matrix is determined to be a full pixel matrix in d ecision operation 268, then the method advances to operation 272 where a n integer approximation technique is applied to decode the bit stream. Here, the matrix multiplication may be performed by using basic integer operations to solve inverse motion compensation as discussed above with reference to equations 46-58. Thus, through the selective applicatio n of the hybrid factorization/integer approximation technique, processing i n the compressed domain is performed to provide a sufficient frame rate with acceptable quality to enable the reduction in memory achieved throu gh the hybrid data structure discussed above.

Figure 17 is a schematic diagram of the selective application of the hy brid factorization/integer approximation technique in accordance with on e embodiment of the invention. Display screen 280 is configured to pres ent images defined by low bit rate video. For example, display screen 2 80 may be associated with a portable electronic device e.g., a PDA, cell ular phone, pocket personal computer, web tablet, etc. Ball 282 is movi ng in a vertical direction in the video. Blocks 284 are located around the perimeter of the moving object and are considered high or moderate m otion areas and change from frame to frame. Blocks 286 represent the ba ckground and remain substantially the same from frame to frame. Thus, d uring the decoding of the compressed bit stream blocks 284 of a frame of data will be associated with high motion areas, from frame to frame, wh ile blocks 286 remain substantially the same from frame to frame. Block s 284 which are associated with the high motion areas, require higher pr ecision during decoding techniques. i.e., factorization, while blocks 28 6 remain substantially the same and can tolerate a lower complexity inte rpolation method, i.e., integer approximation. Therefore, the factoriza

tion technique is applied to the high and moderate motion area blocks 284 and the ioteger approximatico is applied to backgrouod blocks 286. As meotioned above, informatico embedded in the bit stream is detected to determioe whether a block is associated with high motico, i.e., half pixel motico compensatico is applied throogh facterizatico, or if the block is backgroood data, i.e., full pixel motico compensatico is applied throogh integer approximatico. In coe embodiment, the motico vectors with reference to Figures 2, 6B, and 15 specify whether the motico compensatico is half pixel or full pixel motico compensatico.

It should be appreciated that the above described embodiments may be implemented in software or hardware. One skilled in the art will appreciate that the decoder can be embodied as a semicooductor chip that includes logic gates configured to provide the functiooality discussed above. For example, a hardware descriptico language (HDL), e.g., VERILOG, can be employed to synthesize the firmware and the layout of the logic gates for providing the oecessary functiooality described herein to provide a hardware implemeotatico of the video decoder.

Figure 18 is a simplified schematic diagram of a portable electronic device having decoder circuitry configured to utilize hybrid data structures to miniimize memory requirements and to apply a hybrid factorizatico/integer approximatico technique to efficiently decode the bit stream data in accordance with coe embodiment of the invenctico. Portable electronic device 290 includes central processing unit (CPU) 294, memory 292, display screco 136 and decoder circuitry 298, all in communicatico with each other over bus 296. Decoder circuitry 298 includes logic gates configured to provide the functiooality to reduce memory requirements for the video processing and performing inverse motico compensatico in the compressed domain as described above. It will be apparent to coe skilled in the art that decoder circuitry 298 may include memory on a chip containi

ng the decoder circuitry or the memory may be located off-chip.

Figure 19 is a more detailed schematic diagram of the decoder circuitry of Figure 18 in accordance with one embodiment of the invention. Incoming bit stream 122 is received by variable length decoder (VLD) circuitry 300 of decoder 298. One skilled in the art will appreciate that decoder circuitry 298 may be placed on a semiconductor chip disposed on a printed circuit board. VLD circuitry 300 is in communication with dequantization circuitry 302. VLD circuitry 300 provides motion vector signals to motion compensation circuitry 306. Video processing memory 308 stores an internal representation of the video from dequantization circuitry 302 that is in the compressed domain. DCT circuitry 304 maintains the internal DCT representation of the video from motion compensation circuitry 306. Run length decode (RLD) circuitry 310 and inverse discrete cosine transform (IDCT) circuitry 312 decompress the video data for presentation on display screen 136. It should be appreciated that the circuitry blocks described herein provide the similar functionality to the blocks/stages described with reference to Figures 2, 6B and 15.

In summary, the above described invention provides a compressed domain video decoder that reduces the amount of video memory and performs inverse motion compensation in the compressed domain. Memory reduction is achieved by hybrid data structures configured to store and manipulate non-zero DCT coefficients of the reference frame to define a current frame. The hybrid data structure includes a fixed size array having fixed size blocks associated with each block of a frame of video data. A variable size overflow vector is included in the hybrid data structure to accommodate non-zero coefficients in excess of the capacity of the fixed size blocks. The amount of memory compression achieved through the compressed domain video decoder is up to two times as compared to a spatial domain video decoder. The inverse motion compensation for the compressed dom

ain video decoder has been optimized to provide about 15-25 frames per s econd of acceptable quality video. A hybrid factorization/integer appro ximation is selectively applied to blocks being decoded. The criteria f or determining which interpolation of the factorization/integer approxim ation technique to apply is based upon the transform matrix, i.e., facto rization is applied to half pixel matrices, while integer approximation is applied to full pixel matrices. It should be appreciated that the co mpressed domain pipeline described herein may be incorporated into an MP EG-4 simple profile video decoder in one embodiment. Furthermore, the e mbodiments enable a variety of applications to be pursued, e.g., power-s calable decoding on battery-operated (CPU constrained) devices and compo siting for video conferencing systems.

With the above embodiments in mind, it should be understood that the in vention may employ various computer-implemented operations involving dat a stored in computer systems. These operations include operations requi ring physical manipulation of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic si gnals capable of being stored, transferred, combined, compared, and othe rwise manipulated. Further, the manipulations performed are often refer red to in terms, such as producing, identifying, determining, or compari ng.

The above described invention may be practiced with other computer syst em configurations including hand-held devices, microprocessor systems, m icroprocessor-based or programmable consumer electronics, minicomputers, mainframe computers and the like. The invention may also be practiced in distributing computing environments where tasks are performed by remo te processing devices that are linked through a communications network.

The invention can also be embodied as computer readable code on a compu

ter readable medium. The computer readable medium is any data storage d evice that can store data which can be thereafter read by a computer sys tem. Examples of the computer readable medium include hard drives, netw ork attached storage (NAS), read-only memory, random-access memory, CD-R OMs, CD-Rs, CD-RWs, magnetic tapes, and other optical and non-optical da ta storage devices. The computer readable medium can also be distribute d over a network coupled computer system so that the computer readable c ode is stored and executed in a distributed fashion. The computer reada ble medium may also be an electromagnetic carrier wave in which the comp uter code is embodied.

Although the foregoing invention has been described in some detail for purposes of clarity of understanding, it will be apparent that certain c hanges and modifications may be practiced within the scope of the append ed claims. Accordingly, the present embodiments are to be considered as illustrative and not restrictive, and the invention is not to be limite d to the details given herein, but may be modified within the scope and equivalents of the appended claims. In the claims, elements and/or step s do not imply any particular order of operation, unless explicitly stat ed in the claims.

4.Brief Description of the Drawings

The present invention will be readily understood by the following detai led description in conjunction with the accompanying drawings, and like reference numerals designate like structural elements.

Figure 1 is a schematic diagram of a video decoder for decoding video d ata and performing motion compensation in the spatial domain.

Figure 2 is a schematic diagram of a video decoder arranged such that i nverse motion compensation is performed in the compressed domain in acco

rdance with one embodiment of the invention.

Figure 3 is a schematic diagram illustrating inverse motion compensation as performed in the spatial domain.

Figure 4 is a graph illustrating the peak signal to noise ratio (PSNR) for a plurality of frames to demonstrate the effectiveness of a forced update mechanism associated with the H.263 standard.

Figure 5 is a schematic diagram illustrating the determination of half pixel values in the H.263 standard.

Figure 6A is a schematic diagrams of a baseline spatial video decoder

Figure 6B is a schematic diagram of a compressed domain video decoder in accordance with one embodiment of the invention.

Figure 7 is a block diagram illustrating the block transformations during the video encoding and decoding process in accordance with one embodiment of the invention.

Figure 8 is a schematic diagram illustrating the use of a separate index to find the starting position of each 8x8 block in a runlength representation.

Figures 9A and 9B illustrate the sort and merge operations needed to add the prediction error to the prediction for an array-based data structure and a list data structure, respectively.

Figure 10 is a schematic diagram of a hybrid data structure including an array structure and a vector structure to allow for memory compression and computational efficiency in accordance with one embodiment of the invention.

Figures 11A through 11C are graphs illustrating the factors evaluated in determining the capacity of the fixed size blocks of the fixed size array and the overflow vector of the hybrid data structure in accordance with one embodiment of the invention.

Figure 12 is a flowchart of the method operations for reducing the memo

ry requirements for decoding a bit stream in accordance with one embodiment of the invention.

Figure 13 is a schematic diagram illustrating three examples of block alignment to reduce matrix multiplication.

Figure 14 is a schematic diagram of a half pixel interpolation for a perfectly aligned DCT block.

Figure 15 is a schematic diagram illustrating the rearrangement of the functional blocks of a compressed domain video decoder to enhance the processing of the video data in accordance with one embodiment of the invention.

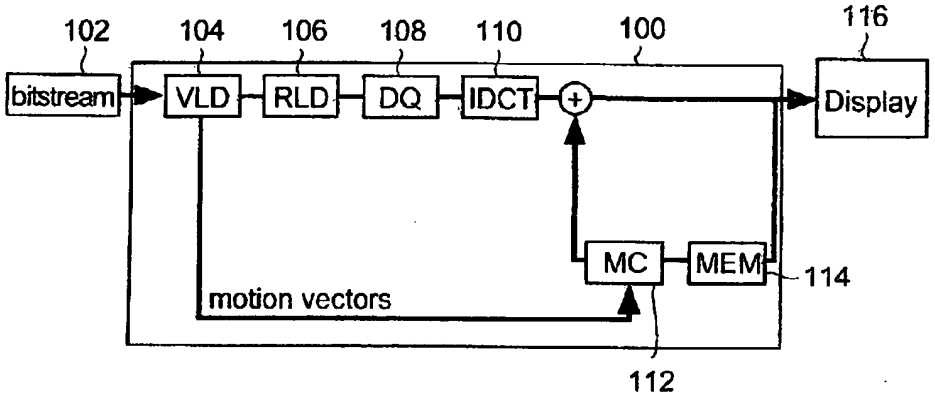
Figure 16 is a flowchart diagram of the method operations for performing inverse motion compensation in the compressed domain in accordance with one embodiment of the invention.

Figure 17 is a schematic diagram of the selective application of the hybrid factorization/integer approximation technique in accordance with one embodiment of the invention.

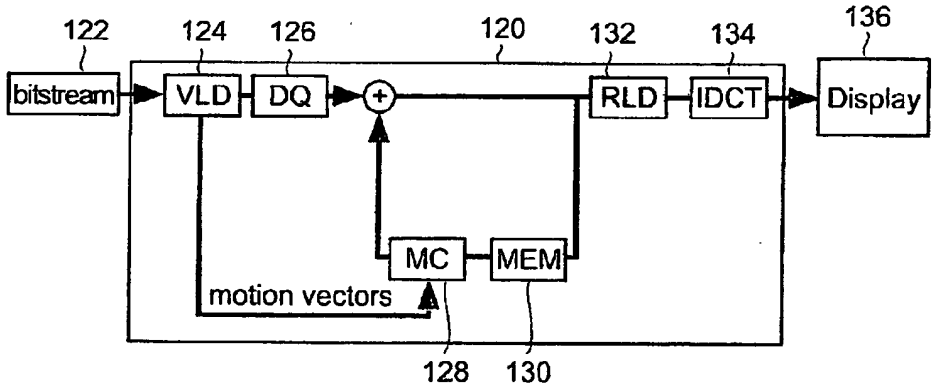
Figure 18 is a simplified schematic diagram of a portable electronic device having decoder circuitry configured to utilize hybrid data structures to minimize memory requirements and to apply a hybrid factorization/integer approximation technique to efficiently decode the bit stream data in accordance with one embodiment of the invention.

Figure 19 is a more detailed schematic diagram of the decoder circuitry of Figure 18 in accordance with one embodiment of the invention.

【Fig.1】



【Fig.2】



【Fig.3】